DTIC
8249-EN-01

Conditional Estimation of Vector Patterns in Remote Sensing and GIS

Interim Report 3a

Principal investigator: Dr. J.M.F. Masuch

1 September 1998

United States Army
European Research Office of the U.S. Army

USARDSG-UK, Edison House
223 Old Marylebone Road
London, NW1 5TH
England

19981006 046

Contract Number: N68171-97 C 9027

CCSOM/Applied Logic Laboratory
PSCW - Universiteit van Amsterdam
Sarphatistraat 143
1018 GD AMSTERDAM
The Netherlands
tel: #.31.20.525 28 52
fax: #.31.20.525 28 00
e-mail: ccsoff@ccsom.uva.nl

R&D 8249-EN-01
Broad Area Announcement Proposal
submitted to the
Remote Sensing / GIS Center USACRREL
72 Lyme Road
Hanover, New Hampshire 03755 USA

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED

4. TITLE AND SUBTITLE
Conditional Estimation of Vector Patterns in Remote Sensing and GIS

5. FUNDING NUMBERS
N68171 97 C 9027

6. AUTHOR(S)
Dr.J.M.F.Masuch

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
CCSOM / PSCW / University of Amsterdam
Sarphatistraat 143
1018 GD AMSTERDAM NL.

8. PERFORMING ORGANIZATION REPORT NUMBER
BAA III 98-2

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION/AVAILABILITY STATEMENT

12b. DISTRIBUTION CODE

13. ABSTRACT (Maximum 200 words)

Within this interim report, we examine specific algorithms for the automatic conversion of raster data into an equivalent co-registered vector format. The focus of this effort is toward automated techniques for the geometric conversion of raster data using high precision edge detection. This investigation will initially focus on perpendicular connections (Section 3), and will then examine techniques for combining raster patterns (Section 4). Algorithms for automated conversion of raster data into vector models will be provided within the discussion of each topical area.

14. SUBJECT TERMS
Remote Sensing, Statistics, Artificial Intelligence, Neural Networks

15. NUMBER OF PAGES
69

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified		

R&D 8249-EN-01

1. Title

Conditional Estimation of Vector Patterns in Remote Sensing and GIS:
Interim Report 3a

2. Abstract

The application of GIS and CAD for the creation, production, and testing of spatial data has increased rapidly throughout the Corps of Engineers. Within this context, raster information from satellite and airborne scanners have been used to provide detailed information for regions within the domestic U.S. that are either poorly documented (with existing vector data) or incomplete with respect to specific spatial attributes. The use of raster data within GIS has been primarily applied to GIS using classification (supervised or un-supervised) as a method to reduce and synthesize the multiple bands of co-registered data. As discussed within ERO Interim Report I and Report II, numerous problems may result in the direct application of classified data. First, this information may be incorrectly clustered due to poor alignment of the data with respect to underlying assumptions (normality, symmetry, bias). Second, the information remains in a purely raster format that requires physical conversion from pixel to line orientation. This conversion is generally accomplished using either "on-screen" digitizing or table-format digitizing. The on-screen method is decidedly poor for high precision applications since the maximum resolution of the resultant vector image is equivalent to the maximum resolution of the display screen (Williams [1998]). The table format method is acceptable, if Quality Control and Quality Assurance (QA/QC) methods are applied at each step in the conversion process (Barr [1994]). Using either on-screen digitizing methods or table format digitizing methods is difficult with satellite and airborne data. This is primarily due to the fact that multiple channels of data are available with very detailed geometric shapes, forms, and patterns. As a result, the digitizing process is laborious and requires careful editing of converted information within a separate QA/QC process. Within this interim report, we examine specific algorithms for the automatic conversion of raster data into an equivalent co-registered vector format. The focus of this effort is toward automated techniques for the geometric conversion of raster data using high precision edge detection. This investigation will initially focus on perpendicular connections (Section 3), and will then examine techniques for combining raster patterns (Section 4). Algorithms for automated conversion of raster data into vector models will be provided within the discussion of each topical area.

3. Perpendicular Connections (Lines and Nodes)

A fundamental property of CAD and GIS is that straight-line junctions and their interconnections contain all information that is needed about the graphical parts of the raster image (starting data). Moreover, right angle turns can be neglected if we are interested only in the logical interpretation of the schematic (Horn [1997]). Thus, the features to be detected are T-junctions, X-junctions, endings, and their interconnections. The connectivity between the GIS features is found by propagation along the connected lines.

The T-junction has four possible orientations:

$$\dashv \quad \perp \quad \vdash \quad \top$$

Figure (1): Perpendicular T- Junctions. Two line segments that cross to form a simple geometric shape. Each T-junction is rotated 90 degrees.

The orientation facilitates the combination of T-junctions into separate components. The X-junction is a crossing of two paths (signals). It is converted into four endings, which are pair-wise entered in a table. The only reason for this is to avoid erroneous connections of the other features when the interconnections are checked. Otherwise, the X-junctions could be neglected completely.

There are two types of endings: end points and border points. An end point can have four orientations. The orientation indicates where to possibly find characters naming a signal. The border points of consecutive windows are associated with each other to connect the separate windows.

The junctions, endings and their interconnections constitute a topological net with four node types. The reduction of a window from lines and junctions to a topological net is illustrated in Figure (2a,b). In Figure (2a), the basic mesh is shown for a series of perpendicular intersections such as those generated within a standard CAD application. In Figure (2b), a primary topological net is shown for the same image consisting of both nodes and line segments. In this illustration, the four main topological nodes are labeled (T-nodes, X-nodes, End-Nodes, and Border Nodes). T-nodes connect perpendicular line segments that

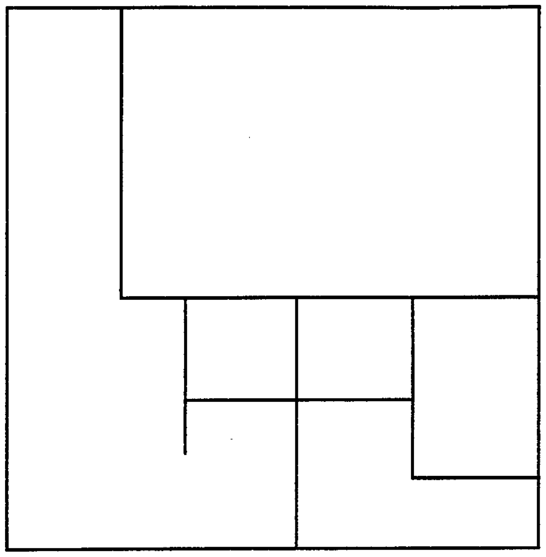

Figure (2a): Perpendicular Line Segments. This figure illustrates a typical sub-area within a standard CAD or GIS AM/FM drawing such as those used to describe utility corridors of transportation paths.

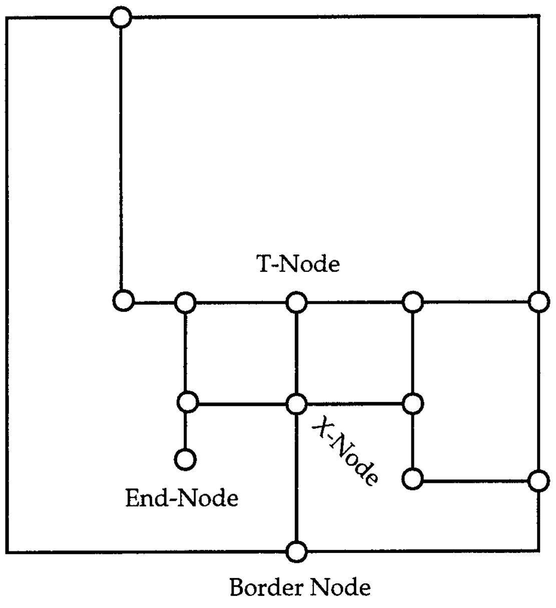

Figure (2b): Example Topology. This figure illustrates a typical topological structure for encoding the perpendicular line segments shown in Figure (2a). For the purpose of this discussion, only four nodes are labeled.

occur within the interior of the raster image. X-nodes connect two crossing line segments that form right angles at the crossing connection. End-nodes mark the conclusion of a line segment within the interior of the raster image. Border nodes mark the conclusion of a line segment along the maximum exterior of the raster image.

The schematic is ideally a two-level raster image: line and background being black and white, respectively. However, several factors contribute to the creation of a Grey-scale image. The main factors are varying illumination, non-uniform sensitivity (shading), and noise.

3.1 Thresholding and Shrinking Geometric Patterns

The first operation (in raster to vector conversion) must be to regain a binary image for edge-detection. Because of the shading context in complex raster images (satellite and airborne data), no single threshold works well throughout the image. The following three statements give a clue to a good local threshold algorithm:

- The line segments are 2 or 3 pixels wide
- The line segments are horizontal or vertical
- The contrast between lines and background is fairly constant throughout the image

These observations indicate that a difference operator should be used. The testing of different operator types shows that the logical OR (| |) of two "one-dimensional" LaPlacians (one horizontal and one vertical) gives a robust result. The operator is formally given by:

$$F_{i,j} = T_1(2\Theta_{ij} - \Theta_{i-3,j} - \Theta_{i+3,j}) \mid\mid T_2(2\Theta_{i,j} - \Theta_{i,j-3} - \Theta_{i,j+3}) \qquad (1)$$

Where
T is a threshold operator T(•) mapped into (0,1) and Θ is the average of a 3x3 pixel kernel where the subscript i,j denotes the center (centroid) of the kernel. The LaPlacian operator for this kernel is shown in Figure (2c).

The thresholding is followed by an 8-pixel connectivity preserving the shrinking to only 1-pixel width (final vectorization). This facilitates the processing of lines and junctions. Note that before shrinking, the lines are expanded one pixel to fill gaps and holes. Short lines and free lines connected to the boundary are deleted.

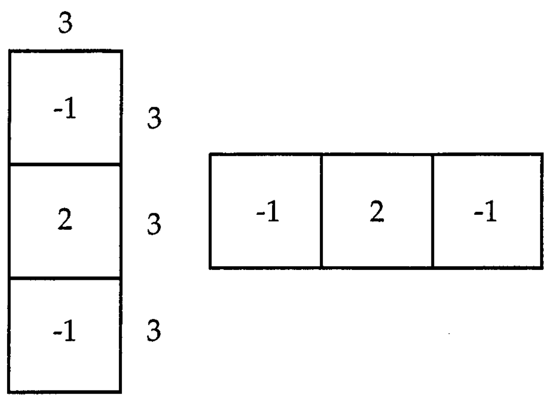

Figure (2c): The two "one-dimensional" LaPlacian Operators. The interior of each pixel (square) is shown with the loading coefficient "-1,2,-1" The exterior numeric digit "3" shows the equalization value (always equal to +3).

3.2 X-Junction Operations

An X-junction can be slightly distorted by the shrinking operation described in Section (3.1). As a result, the geometric shape can be visualized as two T-junctions (see Figure 3). The detection of X-junctions must include detection of closely spaced T-junctions. The T-junctions which are detected by a set of 3x3 templates, are marked and propagated 3 pixels. If two-junctions join, this point is marked as an X-junction. All X-junctions are then propagated four pixels and the fifth pixel in the four step(pixel) expansion is marked. The X-junction is then deleted. To calculate how to combine the four signals, we divide the raster image into four quadrants from the kernel center-point (Figure 4a,b). One end-point should be in each quadrant (if not, a warning is given to the operator). The upper and lower, and left and right pixels are then connected. The X-junction is inserted in a table

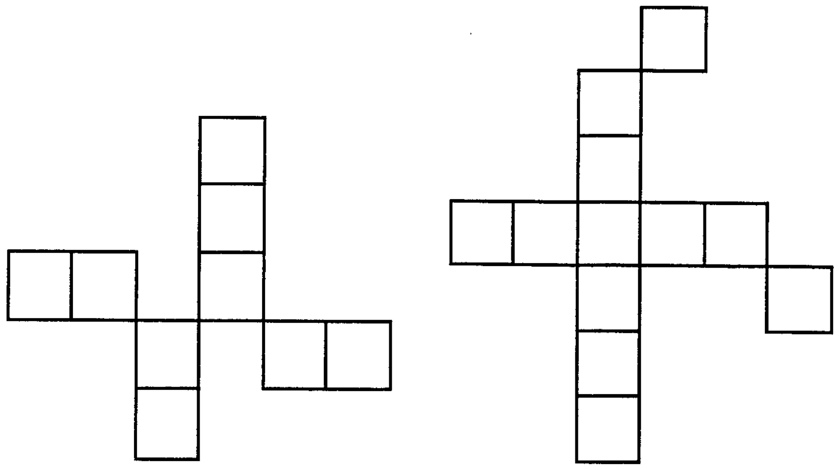

Figure (3): Example X-junctions. In this illustration two sample images are shown with single pixel width geometries. The vectorization of these patterns requires careful detail at intercrossing locations.

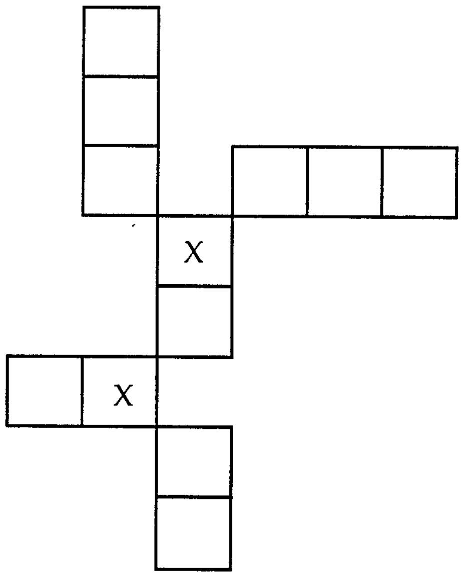

Figure (4a): Elimination of the X-junction Part 1. Pixels adjacent to an X-junction are marked for "shrinkage" prior to formal vectorization.

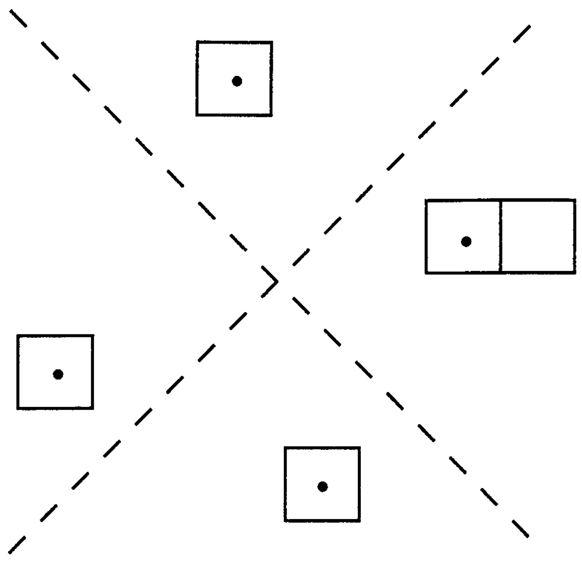

Figure (4b): Elimination of the X-junction Part 2. Tie points "•" are used to separate the junction components into perpendicular quadrants. These quadrants insure that the X-junction (decomposed into separate T-junction elements) can be re-constructed as a proper X geometry.

(Figure 5). The center coordinate and the coordinates for the two combined signal end-points are inserted. In this operation, there is space (adjacent pixels) that is connected to the filled X-junction. These empty pixels can be connected in the final operation.

Center Coordinate	Center Coordinate	Two Combined Points	Connected Junction

x	y	0	x	y	T	x	y	x	y	x	y	T
:	:	:	:	:	:	:	:	:	:	:	:	:

Figure (5): The Connection Table for X-Junctions. This table is used to re-construct the topology for the crossing X geometric forms. The table includes center coordinates for the moving kernel, connected junction points and related labels, the two combined points (x,y), (x,y) and the label for the connected junction.

3.3 End-Points

The orientation of the end-points are determined by propagation and identification of quadrant. The rules and operations governing the assignment of end-points mirror those described in the previous section. The border points are end-points connected to the raster image border (or sub-area delineation). They are temporarily tabulated for connecting the 3x3 kernel windows.

3.4 Connectivity

When all nodes (junctions and end-points) are registered, the interconnections must be deleted. This is accomplished by propagation in the lines (Leonard [1994]). The nodes can not be propagated in parallel, since there is only 4 bits per pixel, and this is not sufficient to assign one unique marker to each node. For this reason, we start with T-nodes, since this allows three segments to be propagated (each leg of the T-junction). A marker is assigned to the node, and the marker is propagated until another node is reached. When the new node is encountered, the coordinate and type of node is inserted in the topology table shown in Figure (5). The coordinate of the T-junction is inserted in the table of the encountered node. Thus, we have a double linked list representing the graph. A line that has been propagated is deleted. When all T-nodes have been propagated, the algorithm continues with the other nodes until no more lines remain in the kernel window.

Propagation is a rather time consuming operation. Each iteration requires approximately 20 milliseconds on a Sun Sparc 20 single processor system. Thus, a propagation from one side of the kernel window to the other takes 64 x 20 milliseconds = 1.28 seconds. The time for propagation is reduced by "sequential" propagation, which implies that lines going to the right or down from the node are propagated in one operation. This would, on the average, reduce the time 50 percent (half the lines are going down or to the right from the nodes). Since the nodes in the upper left corner are propagated first, the reduction is larger as the kernel moves downward through the raster image.

4. Combining Raster Patterns

Raster patterns are combined using arithmetic operations for addition, subtraction, multiplication, and division. Each operation is selected based upon propagation. When a segment is propagated the RGB signatures are combined to create an optimal connection that "feathers" identical features with minimum distortion. When performing the pixel addition, the following algorithm is applied:

$$\begin{aligned} p_{i,j}(r) &= p_{i,j}(\alpha) + p_{i,j}(\varsigma) \quad ; && 0 \le p_{i,j}(r) \le p_{i,j}(m) \qquad (2) \\ p_{i,j}(r) &= p_{i,j}(m) && ;\ \text{otherwise} \end{aligned}$$

Where

$p_{i,j}(r)$ is the pixel value assigned to the result (or target) window r at the horizontal position i and vertical position j within the original raster image.

$p_{i,j}(\alpha)$ is the pixel value extracted from the base image at the horizontal position i and vertical position j.

$p_{i,j}(\varsigma)$ is the pixel value of the kernel window at the horizontal position i and vertical position j, and

$p_{i,j}(m)$ is the maximum value any pixel can attain at the horizontal position i and vertical position j. For 8 bit color $p_{i,j}(max) = 255$, for 24 bit color the maximum is scaled relative to the 16.7 million colors.

Pixels are added together provided the result does not exceed the maximum displayable intensity produced by the image processor.

When performing the pixel subtraction, the following algorithm is applied:

$$\begin{aligned} p_{i,j}(r) &= p_{i,j}(\alpha) - p_{i,j}(\varsigma) \quad ; && 0 \le p_{i,j}(r) \le p_{i,j}(m) \\ p_{i,j}(r) &= 0 && ;\ p_{i,j}(\alpha) < p_{i,j}(\varsigma) \qquad (3) \\ p_{i,j}(r) &= p_{i,j}(m) && ;\ \text{otherwise} \end{aligned}$$

Where

$p_{i,j}(r)$, $p_{i,j}(\alpha)$, $p_{i,j}(\varsigma)$, and $p_{i,j}(m)$ are as described in Equation (2).

In Equation (3), the algorithm assigns a pixel result $p_{i,j}$ (r) equal to zero when the magnitude of the copied pixel $p_{i,j}$ (α) is less than or equal to the magnitude of the kernel pixel $p_{i,j}$ (ς). This constraint is required to insure that no pixel assignments are made that result in negative digital values. In a similar manner, the pixel result $p_{i,j}$ (r) is set equal to the maximum level for the video display $p_{i,j}$ (m) when the result $p_{i,j}$ (r) is greater than or equal to the maximum pixel depth.

When performing the pixel multiplication, the following algorithm is applied:

$$p_{i,j}(r) = p_{i,j}(\alpha) * p_{i,j}(\varsigma) \;;\; 0 \le p_{i,j}(r) \le p_{i,j}(m) \qquad (4)$$

$$p_{i,j}(r) = p_{i,j}(m) \qquad ;\text{otherwise}$$

Where
$p_{i,j}$ (r), $p_{i,j}$ (α), $p_{i,j}$ (ς), and
$p_{i,j}$ (m) are as described in Equation (2).

As shown in Equation (4), pixels are multiplied at a given horizontal (i) and vertical location (j) provided the result $p_{i,j}$ (r) is bounded between zero and the maximum displayable intensity level $p_{i,j}$ (m). When the result $p_{i,j}$ (r) exceeds the maximum $p_{i,j}$ (m), the target image is assigned a maximum level that can be displayed by the workstation. Pixel division is performed based upon the respective digital values found within the raster pattern and kernel windows. The division is rounded to the nearest integer value for display and is bounded within the minimum and maximum displayable intensities of the workstation. When performing the pixel division, the following algorithm is applied:

$$
\begin{aligned}
p_{i,j}(r) &= f\{p_{i,j}(\alpha) \div p_{i,j}(\varsigma)\} && ;\ 0 \le p_{i,j}(r) \le p_{i,j}(m)\ ;\ p_{i,j}(\varsigma) > 0 \\
p_{i,j}(r) &= p_{i,j}(m) && ;\ p_{i,j}(\varsigma) = 0 \qquad (5) \\
p_{i,j}(r) &= p_{i,j}(m) && ;\ p_{i,j}(r) > p_{i,j}(m)
\end{aligned}
$$

Where
$p_{i,j}(r)$, $p_{i,j}(\alpha)$, $p_{i,j}(\varsigma)$, and $p_{i,j}(m)$ are as described in Equation (2), and the function f is used to round the result of the division operation to the nearest integer.

In Equation (5), the division is performed provided the result of the operation is bounded between zero and $p_{i,j}(m)$. When the denominator $p_{i,j}(\varsigma)$ is equal to zero, the division cannot be performed, and the resultant pixel is assigned the maximum intensity (i.e. $p_{i,j}(r) = p_{i,j}(m)$). For the condition when $p_{i,j}(r) > p_{i,j}(m)$, the level is scaled down to the maximum displayable intensity of the workstation. When the division is performed, the result is rounded to the nearest integer for representation. For example, if the result of a division is such that $p_{i,j}(\alpha) \div p_{i,j}(\varsigma) = 26.4$, a digital value of 26 is assigned to the pixel $p_{i,j}(r)$. Alternatively, if the result of a division is such that $p_{i,j}(\alpha) \div p_{i,j}(\varsigma) = 26.5$ or 26.6, a digital value of 27 is assigned to the pixel $p_{i,j}(r)$. Note that pixel division is extremely sensitive to the assignment of the source pattern and target (kernel) images, since these assignments control the resultant assignment $p_{i,j}(r) = f\{p_{i,j}(\_C) \div p_{i,j}(\_V)\}$ in Equation (5).5. Connecting Kernel Windows During VectorizationThe nodes and arcs from the 64 x 64 windows are put together to produce the global topological net, which contains the information needed for a logical description of the schematic. The process steps are illustrated in Figure (6a,b). The procedure eliminates all border nodes except those that represent global border nodes (signals connected to the border of the schematic).

The windows are digitized from separate raster patterns, which give a small uncertainty in the location of a node. A small overlap of the windows is needed to ensure that nodes at the window borders will be detected.

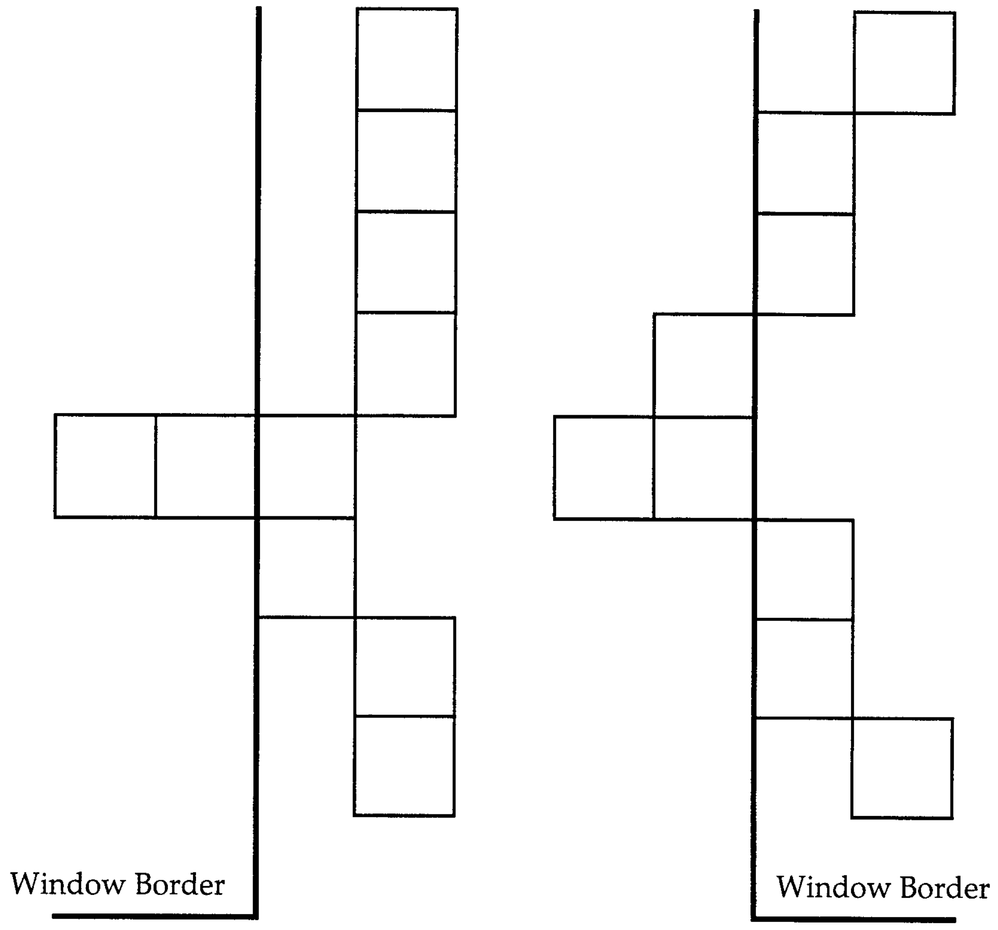

Figure (6a): Digitizing and Shrinking a T-junction. The T-junction is, due to registration error, moved one pixel to the left in the right image. This produces an erroneous interpretation of the vector pattern.

When overlap is introduced, we will not miss any nodes, but a node can be detected in any of the four overlapping windows (kernels). This introduces the problem of identifying corresponding nodes and reducing them into one "common" node with the correct connections. The reduction of the nodes is facilitated if the windows are joined in the sequence shown in Figure (7a).

A solution to the problem with connection of windows would be to use a scanner capable of digitizing the whole sheet. The digitized image could then either be compressed and stored for processing by a computer or by a pipelined processor immediately.

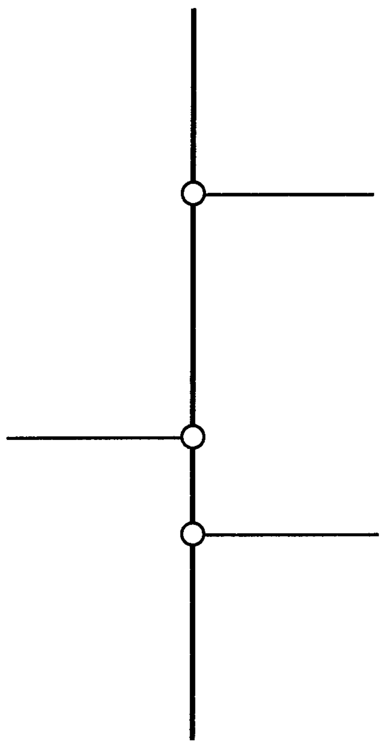

Figure (6b): Vector T-junctions. The raster T-junction shown in Figure (6a) is projected into vector space. One possible discrimination is shown using perpendicular elements.

Only two areas at a time are joined, so there can be only two nodes that correspond to each other. Two nodes are treated as equal if they are within a square of 4x4 pixels and have the same properties.The connections to the nodes are treated pairwise (the corresponding connections from the two nodes are applied). The two corresponding connections (Pixel; "A" and "B"), fulfill one of the following three alternatives shown in Figure (7b,c,d):

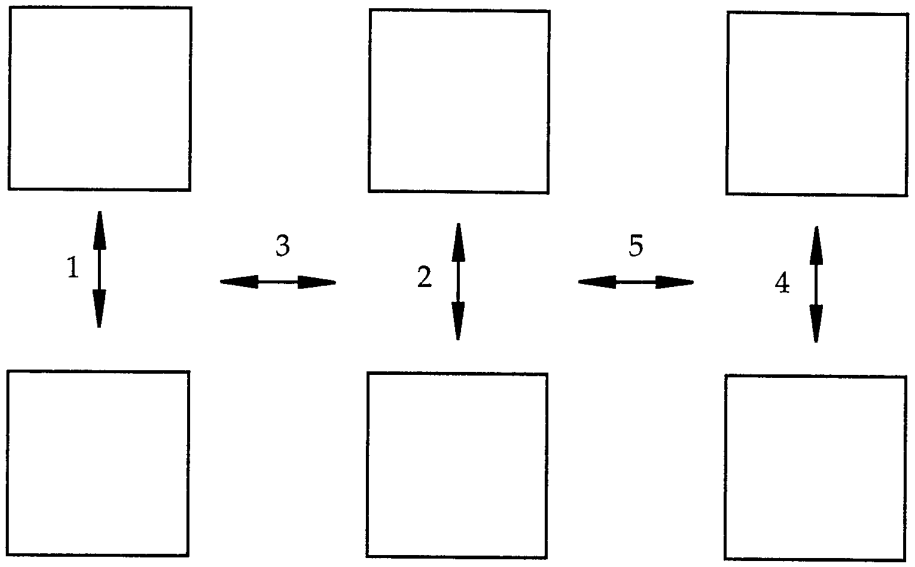

Figure (7a): The Joining of Windows. Pixels are joined from upper-left to lower-right. The joining begins at the far left edge and proceeds in steps as shown.

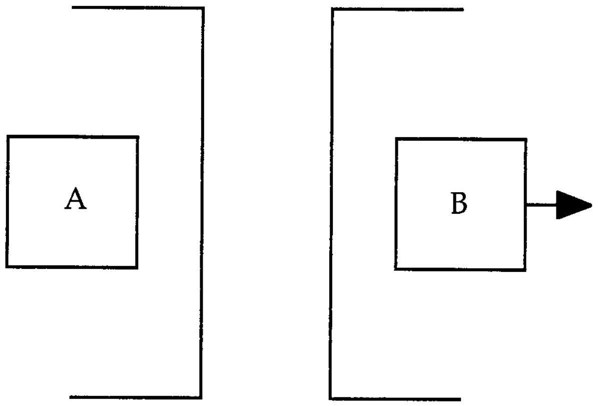

Figure (7b): No Connection Between Adjoining Kernel Windows. Pixel "A" is clearly separated from Pixel "B".

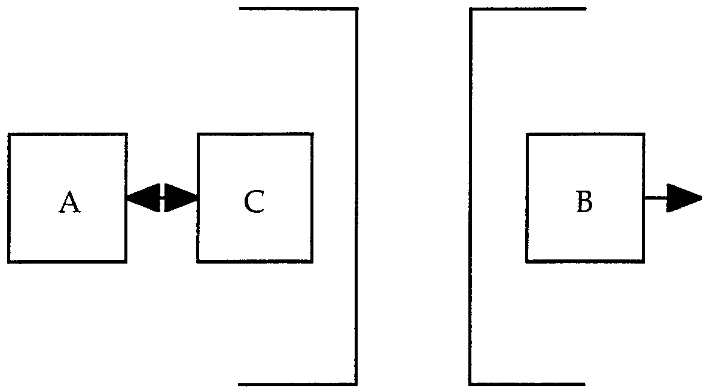

Figure (7c): Pixel "A" is Connected to Pixel "C", and Pixel "B" Remains Unconnected.

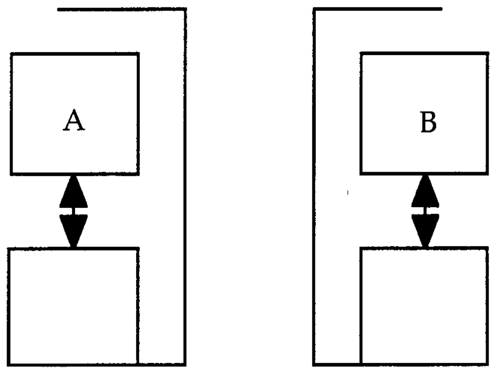

Figure (7d): Pixel "A" and Pixel "B" are Connected to Border Pixels Along the Kernel Window.

• Pixel "A" has no connection, while Pixel "B" is connected to a node which is not a border-node adjacent to Pixel "A").
• Pixel "A" is connected to a border-node (Pixel "C") within the neighborhood of Pixel "B". Pixel "B" is also connected to the border node.
• Both Pixel "A" and Pixel "B" are connected to the two corresponding border nodes.

In all cases (a-c), the algorithm keeps Pixel "A" and deletes Pixel "B". Pixel "A" is connected to the border element. Pixel "C" and all neighborhood border pixels are deleted.

6. Border Connections

Two border nodes that correspond to each other are deleted and the double linked graph is updated (Figure 8a). It is mainly two problems that complicate the elimination of border nodes:There are border nodes which have no corresponding node (Figure 8b).There is a long distance between two corresponding nodes (Figure 8c).The same border node can also, as the other nodes, be present in two separate kernel windows.

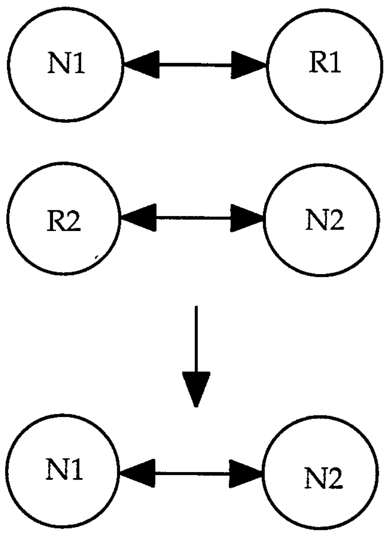

Figure (8a): The Elimination of Pixels within a Linked Graph. Adjacent pixels are labeled and removed based upon a linked list. Each Pixel "P1" and "P2" is linked to a unique node "N1" and "N2". The resultant topology is a linked list of nodes ("N1" and "N2" only).

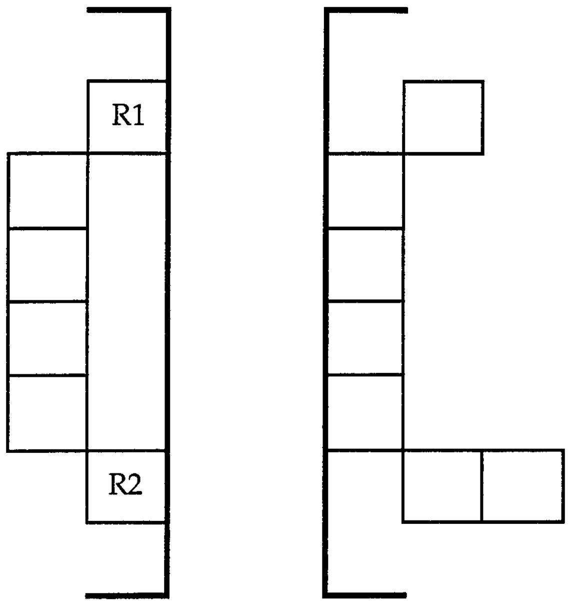

Figure (8b): Border Nodes with No Corresponding Node.

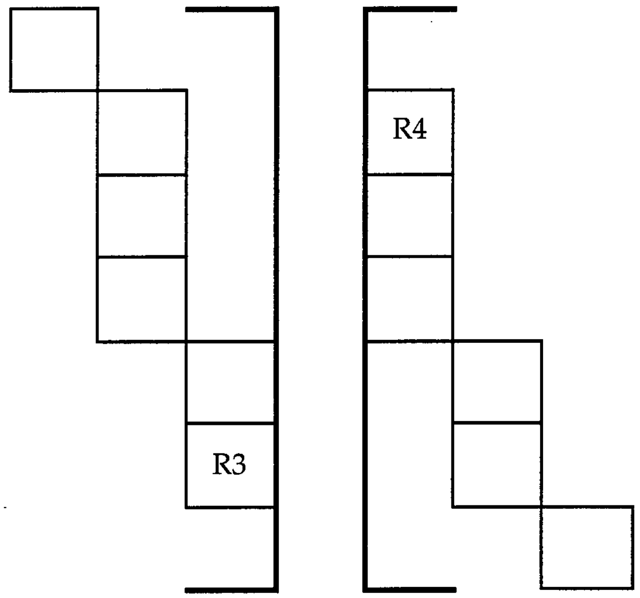

Figure (8c): Distance Separation Between Border Nodes.

The connection of border nodes is directed by the following rules:

• Border nodes can not be joined in a cross pattern as shown in Figure 9a.
• Border nodes can be without corresponding nodes only if they are connected to a border node (for example Figure 9b), or are part of a double-detected node as shown in Figure 9c.
• All border nodes, except the global edge of the complete raster image, are eliminated.

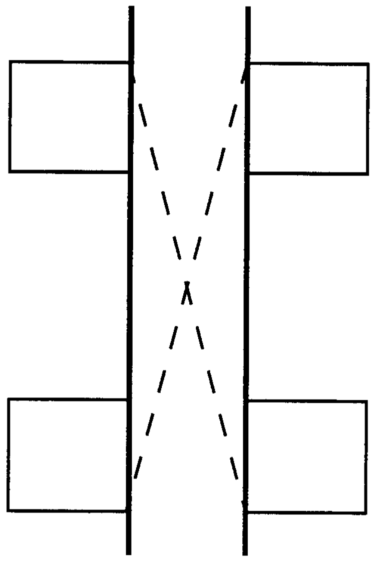

Figure (9a): Pixels Linked within a "Cross" Pattern.

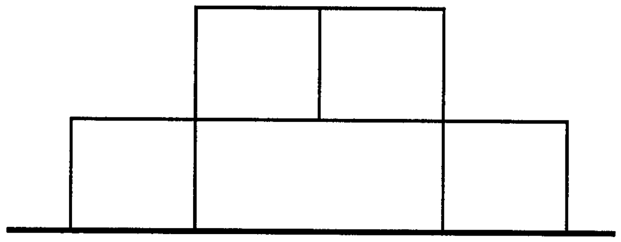

Figure (9b): Pixels Connected to a Common Border.

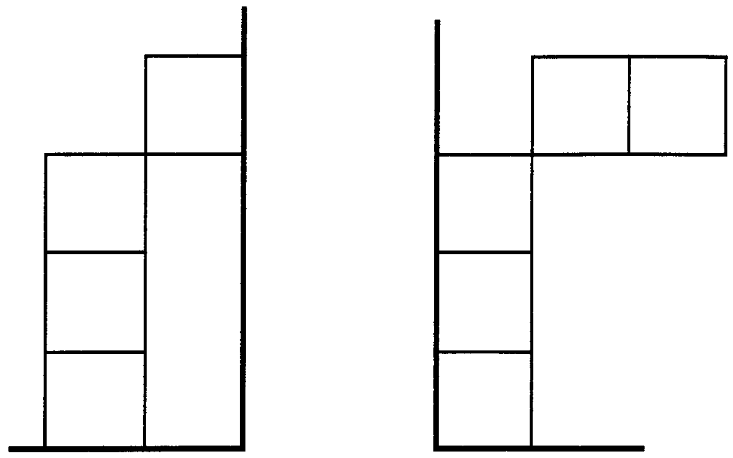

Figure (9c): Pixels Separated by the Moving Kernel Window Resulting in a "Double-Detected" Corner Node.

The routine for connection of the border nodes starts at one end of the borderline and processes the border pixels in a pairwise fashion. If the distance between the nodes is large, then the second rule is used to decide when a node can be neglected. This technique works very well, but there are cases where this is not enough information to decide which border nodes should be neglected (Marr and Hildreth [1997]). Figure 9d shows two borders that should be connected. The lower border has two border nodes more than the upper border. Which nodes should be neglected? Within the basic algorithm, it is impossible to say, because the original raster image can be of the form shown in Figure (9e) or Figure (9f). This example occurs in very limited cases, but is provided to show that the basic connection routine can make errors. To improve the algorithm, one must also use the orientation of the border points. When all the kernel windows have been connected, there is a global graph describing the graphics part of the schematics. This graph is searched to find the components and signals. This produces a new graph, where the nodes are components or logical units and the arcs are signals.

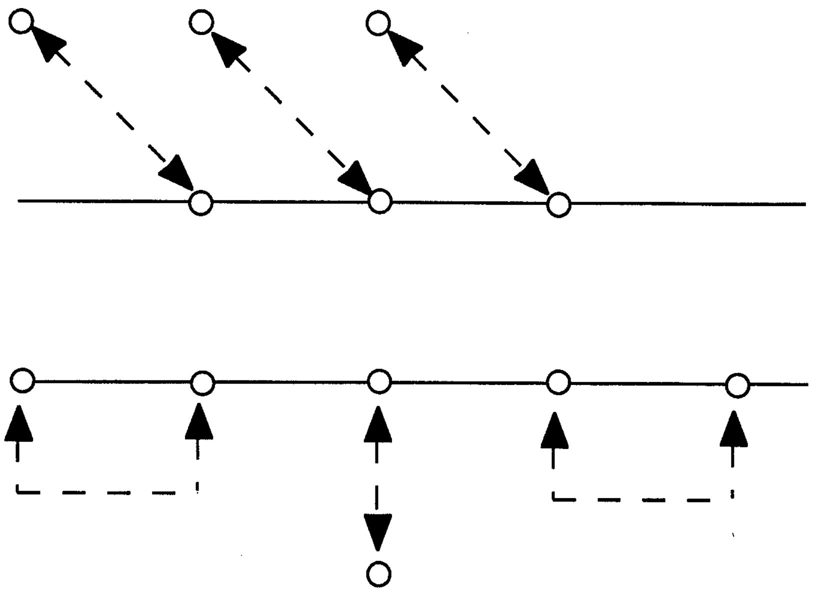

Figure (9d): Single Border Connections.

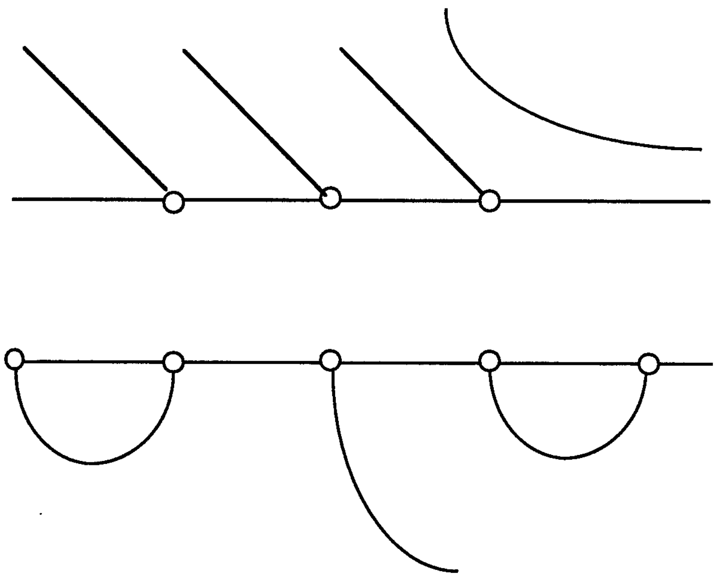

Figure (9e): Connection Path Phase One

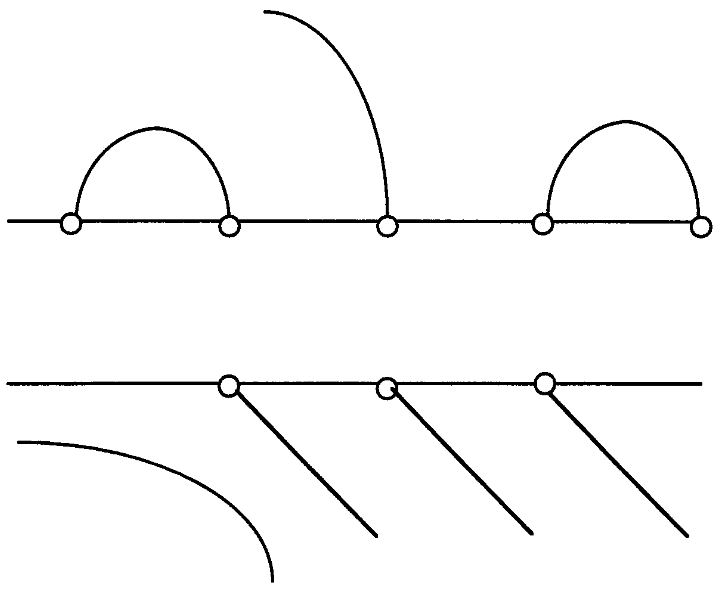

Figure (9f): Connection Path Phase Two (180-Degree Rotation).

7. Conclusions

Due to the drawing conventions adopted, a component is characterized as a geometric square with connections on two-opposing sides. Hence, a component can be recognized by a search in the T-node table, where it is represented as a cycle of T-nodes. Other conditions that must be fulfilled by the T-nodes are:

1) The cycle must contain two sequences of similar angles of orientation.
2) The orientations must differ by π.
3) The "legs" of the T-shaped patterns must point outward from the component.

Only a component can fulfill these requirements in a correctly drawn schematic.

When the components have been found, each element is assigned a digital number and four coordinates describing its relative position. The connections are assigned digital numbers according to their respective location at the connection node. The remaining

When the components have been found, each element is assigned a digital number and four coordinates describing its relative position. The connections are assigned digital numbers according to their respective location at the connection node. The remaining part of the graph is signal (non-noise separation). Each signal is a binary tree. A binary search is used to find the connection between the components. Signals that have end points are associated with a neighborhood in connection with the end point.

In a subsequent procedure, the interior of the components and the signal neighborhood are rescanned for recognition of component identification and signal names. The orientation of the end points and components indicates the orientation of the character strings, which facilitates a correct rescanning of the raster pattern. The rescanning is done at a high resolution of approximately 2.2 pixels per millimeter.

The algorithm described within this research demonstrates a viable alternative to standard digitizing of raster patterns into a vector model. Whether an automatic vectorization method can provide benefits greater than costs depends on the system processing capabilities and the reliability of the output vector model. The equipment costs for interactive systems and for a system of the proposed type are comparable, but the programming is more complex in the latter case (automated vector modeling). The economic advantage of the automatic conversion stems from its shorter processing time which allows engineers to be more effective. One raster scene at 24-bit pixel depth (640x480 pixels) can be processed in about two minutes. Depending upon pattern complexity and classification, an equivalent vectorization using hand-digitizing will require from 2 to 4 hours with additional error in interpretation (non-systematic approach). The algorithm has proven to give reliable results, which indicates that the pattern-recognition process is not a limiting factor. In future research, the algorithm will be adapted to non-grayscale images. This will require clear separation of red, green, and blue components as separate bands of data, as opposed to linked tables. This may be easily accomplished using the covariance matrix as a scaled weight between spectral bands. In this approach the covariance or eigenvalue is used as a digital multiplier for connecting the pattern across spectral bands. This approach holds genuine promise for applications where basic patterns cannot be synthesized (reduced) to an orderly grayscale representation. In this case, the vectorization can be applied to a non-classified image with clear separation based upon the "eigenvectors" that connect each spectral band. Additional research is warranted in this area for the application of automated methods with multispectral data.

8. References

Barr, A., Global and Local deformations of Solid Primatives, J. Comput. Graphics, 18, 1994.

Horn, B. K. P., Understanding Image Intensities, J. Artif. Intell., 8, 1997.

Leonard, G., Database for Speaker-Independent Digit Recognition, Proceedings IEEE Int. Conf. Acoustics, Speech Signal Process, San Diego, Mar. 1994.

Marr, D. and E. Hildreth, Theory of Edge Detection, Proc. R. Soc. London, 207, 1997.

Williams, L., Pyramidal Parametrics, , J. Comput. Graphics, 22, 1998.

9. Appendix

Within this appendix, fully documented source code is provided for all vector conversion algorithms. Additional algorithms are provided for the estimated of variance within vector fields (topographic tables). The vector fields are used to estimate the mean representation for the final vector pattern. The variance estimation procedures are required for quality assurance and quality control of all algorithms. The complete simulation engine is programmed for Windows 95 and Solaris UNIX applications.

```
/*************************************************************************
Vector CApplication.c


The Abstract class for a UNIX application. Programmers must create a subclass of
CApplication for their own programs. The application is at the top of the chain of
command and defines the behavior of the program and its interaction with the UNIX
OS.

      SUPERCLASS = CVectorization

 ************************************************************************/


#include "Global.h"
#include "Constants.h"
#include "CApplication.h"
#include "Commands.h"
#include "OSChecks.h"
#include "CDirector.h"
#include "CError.h"
#include "TBUtilities.h"
#include "CDesktop.h"
#include "CBartender.h"
#include "CClipboard.h"
#include "CDecorator.h"

/*** Global Variables ***/

extern CApplication      *gApplication;     /* Application object              */
extern CDesktop          *gDesktop;         /* The visible Desktop             */
extern CBartender        *gBartender;       /* Manages all GUIs                */
extern CClipboard        *gClipboard;       /* Copies and Pastes               */
extern CVectorization    *gGopher;          /* get commands                    */
extern CError            *gError;           /* Error handler                   */
extern CDecorator        *gDecorator;       /* Decorator for arr. windows      */
extern long              gSleepTime;        /* Max time between event          */
extern Boolean           gHasWNE;           /* Is WaitNextEvent implemented?   */
extern Boolean           gInBackground;     /* In background under MultiSpectral */
extern OSType            gSignature;        /* Creator for Application's files */
extern EventRecord       gLastMouseDown;    /* Previous mouse down event       */
extern EventRecord       gLastMouseUp;      /* Previous mouse up event         */
extern CView             *gLastViewHit;     /* Last view clicked in            */
extern short            gClicks;           /* Click counter, = 1 single click  */
                                           /*                = 2 double click  */
                                           /*    etc.                          */
extern CursHandle       gIBeamCursor;      /* I-beam for text views            */
extern CursHandle       gWatchCursor;      /* Watch cursor for waiting         */
extern RgnHandle        gUtilRgn;          /* Utility region                   */



/*** Class Constants ***/

#define          GROW_FAILURE      0L      /* Return codes from GrowZoneFunc   */

#define          GROW_SUCCESS      1L

#define          JUMPBUFFER_A1     5       /* Index of A1 in JumpBuffer        */



/**** V E C T O R   I N I T I A L I Z A T I O N   M E T H O D S ****/

/**************************************************************************
 IApplication

           Initialize an Application.
 **************************************************************************/

void  CApplication::IApplication(
      short       extraMasters,
                                           /* Number of additional master      */
                                           /*   blocks to allocate             */
      Size        aRainyDayFund,           /* Bytes of memory to reserve       */
                                           /* for a rainy day                  */
      Size        aCreditLimit)            /* Max unapproved memory request    */
{

                                           /* We haven't reached the event     */
                                           /*   loop yet. Flag A1 (jump addr)  */
                                           /*   so we don't try to jump there. */

      eventLoopJump[JUMPBUFFER_A1] = NULL;

      InitToolbox();

                                           /* Initialize Toolbox Managers      */
                                           /* Fine tune the Memory Manager     */

      InitMemory(extraMasters, aRainyDayFund, aCreditLimit);

      /** Instance Variables **/
      itsSwitchboard = new(CSwitchboard);
      itsSwitchboard->ISwitchboard();
	itsDirectors = new(CCluster);
	itsDirectors->ICluster();
	itsIdleChores = new(CList);
	itsIdleChores->IList();
	itsUrgentChores = new(CCluster);
	itsUrgentChores->ICluster();
	urgentsToDo = FALSE;
	running = TRUE;

								/** Global Variables          **/
	gSignature = 'VECT';
	gHasWNE = WNEIsImplemented();
	gSleepTime = 0;
	/* We want an early first Idle*/
	gError = new(CError);


								/* Cursors                    */
	gIBeamCursor = GetCursor(iBeamCursor);
	HNoPurge(gIBeamCursor);
	gWatchCursor = GetCursor(watchCursor);
	HNoPurge(gWatchCursor);

	gUtilRgn = NewRgn();

	MakeDesktop();
	MakeClipboard();
	MakeDecorator();
	SetUpFileParameters();
	SetUpGUIs();

	gGopher = this;
	gLastViewHit = NULL;
	gLastMouseUp.when = 0L;
	gClicks = 0;
}

/*****************************************************************************
 InitToolbox

		Initialize the "Vector" UNIX Toolbox
 *****************************************************************************/

void	CApplication::InitToolbox()
{
	InitGraf(&thePort);
							/* Standard initialization calls    */
	InitFonts();
	InitWindows();
	InitGUIs();
	TEInit();
	InitDialogs(NULL);
							/* ??? Add a ResumeProc             */
	InitCursor();
}.
```

```
/*****************************************************************************
 InitMemory

           Initialize the Heap and create extra master pointer blocks
 *****************************************************************************/

void  CApplication::InitMemory(
      short       extraMasters,
                                           /* Number of additional master      */
                                           /*   blocks to allocate             */
      Size        aRainyDayFund,
                                           /*   a rainy day                    */
      Size        aCreditLimit)
                                           /* Max memory request which can be  */
                                           /*   taken from rainyDayFund        */
                                           /*   without prior approval         */
{
      MaxApplZone();
                                           /* Grow heap to maximum size to     */
                                           /*   help prevent fragmentation     */
                            /* Call MoreMasters to allocate extra master pointers.*/
                            /* Experiment to determine how many master pointers   */
                            /* the application uses under session of heavy use.   */

      while (extraMasters-- > 0)
            MoreMasters();

      SetGrowZone(GrowZoneFunc);           /* Traps out of memory conditions   */
      rainyDay = NULL;                     /* Clear rainyDay handle in case    */
                                           /*   allocating the rainy day fund  */
                                           /*   fail! This is really bad.      */


      rainyDayFund = aRainyDayFund;        /* Set instance variables           */
      creditLimit = aCreditLimit;
      rainyDayUsed = FALSE;
      memWarningIssued = FALSE;
      loanApproved = FALSE;
      rainyDay = NewHandle(rainyDayFund); /* Reserve a block of memory        */
}


/*****************************************************************************
 MakeRasterDesktop

           Create the global Desktop object, which is the top level of
           the visual hierarchy. Overrride this method to use a desktop
           which supports floating windows or other extensions to the
           standard desktop.
 *****************************************************************************/
```

```
void  CApplication::MakeDesktop()
{
      gDesktop = new(CDesktop);
                                                /* Use a standard Desktop            */
      gDesktop->IDesktop(this);
}




/******************************************************************************
 MakeClipboard

            Create the global Clipboard object (Raster & Vector Objects)
 ******************************************************************************/


void  CApplication::MakeClipboard()
{
      gClipboard = new(CClipboard);
                                                /* Use standard Clipboard            */
      gClipboard->IClipboard(this, TRUE);
}


/******************************************************************************
 MakeDecorator

            Create the global Decorator object, which controls the
            arrangement of windows on the screen. Override if a
            different Decorator is desired.
 ******************************************************************************/

void  CApplication::MakeDecorator()
{
      gDecorator = new(CDecorator);
                                                /* Use standard Decorator            */
      gDecorator->IDecorator();
}



/******************************************************************************
 SetUpFileParameters

Set values of the parameters used by the Standard File Package.
 ******************************************************************************/

void  CApplication::SetUpFileParameters()
{
```

```
                                             /** Get File Parameters **/
      sfNumTypes = -1;
                                             /* Display all files              */
      sfFileTypes[0] = '????';
                                             /* Just put something here        */
      sfFileFilter = NULL;                   /* No extra filtering of files    */
      sfGetDLOGHook = NULL;                  /* No special handling of items   */
      sfGetDLOGid = getDlgID;                /* Use built-in get dialog box    */
      sfGetDLOGFilter = NULL;                /* No special event processing    */
}


/*****************************************************************************
 SetUpGUIs

           Set up the GUIs used by the application
 *****************************************************************************/

void  CApplication::SetUpGUIs()
{
      gBartender = new(CBartender);
      gBartender->IBartender(MBARapp);
      AddResGUI(GetMHandle(GUIVector), 'DRVR');
      gBartender->SetDimOption(GUIVector, dimNONE);
}




/**** V E C T O R   A C T I O N   M E T H O D S ****/

/*****************************************************************************
 Notify {OVERRIDE}

           A subordinate is notifying the Application that a task has been
completed.
 *****************************************************************************/

void  CApplication::Notify(
Ctask     *theTask)                    /* The completed task                   */
{
}                                      /* Null Method                          */
```

```
/*****************************************************************************
 DoKeyDown {OVERRIDE}

            Respond to a keystroke. Application has no supervisor to whom to pass
the buck. Default action is to ignore keystrokes.
 *****************************************************************************/


void  CApplication::DoKeyDown(
      char              theChar,         /* The associated character        */
      Byte              keyCode,         /* Code for the associated key     */
      EventRecord       *UNIXEvent)      /* Key down event record           */
{
}
                                         /* Null Method                     */



/*****************************************************************************
 DoAutoKey {OVERRIDE}

Respond to a key being held down. By default, keystrokes are ignored.
 *****************************************************************************/


void  CApplication::DoAutoKey(
      char              theChar,         /* The associated character        */
      Byte              keyCode,         /* Code for the associated key     */
      EventRecord       *UNIXEvent)      /* Autokey event record            */
{
}
                                         /* Null Method                     */



/*****************************************************************************
 DoKeyUp {OVERRIDE}

            Respond to a key being released. Usually, these events will be masked
out by the operating system. Ignore them by default.
 *****************************************************************************/


void  CApplication::DoKeyUp(
      char              theChar,         /* The associated character        */
      Byte              keyCode,         /* Code for the associated key     */
      EventRecord       *UNIXEvent)      /* Key up event record             */
{
}
                                         /* Null Method                     */
```

```
/****************************************************************************

 DoCommand

           Handle a command. Application has no supervisor which can handle the
           command. If we don't handle the command here, it will never be
           executed (and that's an error).
 ****************************************************************************/


void  CApplication::DoCommand(
      register long     theCommand)        /* Command number                 */
{
      Str255            theDA;             /* Name of Desk Accessory to open */
      SFReply           UNIXSFReply;       /* Standard File reply record     */

      if (theCommand < 0) {
            if (HiShort(-theCommand) == GUIVector) {
                                          /* Open a DA or launch another     */

                                          /*   application under MultiSpectral*/
            GetItem(GetMHandle(GUIVector), LoShort(-theCommand), theDA);
                  OpenDeskAcc(theDA);
            }
      } else {

            switch (theCommand) {

                  case cmdNew:
                        SetCursor(*gWatchCursor);
                        CreateDocument();
                        break;


                  case cmdOpen:
                        ChooseFile(&UNIXSFReply);
                        if (UNIXSFReply.good) {
                              SetCursor(*gWatchCursor);
                              OpenDocument(&UNIXSFReply);
                        }
                        break;


                  case cmdClose:
                        if (IsSystemWindow((WindowPeek)FrontWindow()))
                              CloseDeskAcc(((WindowPeek)FrontWindow())-
                              >windowKind);
                        break;
                  case cmdQuit:
                        Quit();
                        break;
                        case cmdUndo:
                        case cmdCut:
                        case cmdCopy:
                        case cmdPaste:
                        case cmdClear:
                                SystemEdit((short)(theCommand - cmdUndo));
                                break;


                        case cmdToggleClip:
                                gClipboard->Toggle();
                                break;
                }
        }
}




/*************************************************************************
 UpdateGUIs {OVERRIDE}

                Perform GUI management tasks
 *************************************************************************/


void    CApplication::UpdateGUIs()
{
        Str63 undoStr;

        gBartender->EnableCmd(cmdQuit);

        if (gInBackground) {
                gBartender->EnableCmd(cmdClose);
                gBartender->EnableCmd(cmdUndo);
                gBartender->EnableCmd(cmdCut);
                gBartender->EnableCmd(cmdCopy);
                gBartender->EnableCmd(cmdPaste);
                gBartender->EnableCmd(cmdClear);
        } else {
                gBartender->EnableCmd(cmdToggleClip);
        }

        if (!rainyDayUsed) {
                gBartender->EnableCmd(cmdNew);
                gBartender->EnableCmd(cmdOpen);
        }

        GetIndString(undoStr, STRcommon, strUNDO);
        gBartender->SetCmdText(cmdUndo, undoStr);
}
```

```
/**** M E M O R Y   M A N A G E M E N T ****/


/*****************************************************************************
 RequestMemory

            Turn on/off the flags which indicate whether subsequent memory
            requests can use the rainy day fund and can fail
 *****************************************************************************/


void  CApplication::RequestMemory(
      Boolean           aLoanApproved,
      Boolean           aCanFail)
{
      loanApproved = aLoanApproved;
      canFail = aCanFail;
}



/*****************************************************************************
GrowMemory

A memory request can't be filled by the UNIX Memory Manager. Free up memory if
possible or gracefully shutdown.

A rainy day fund of reserve memory is maintained to monitor low memory conditions.
If the application can't free up enough memory on its own, we use the reserve fund.
The user is notified when we tap into the reserve fund so that he/she can take
steps to free up memory (perhaps by closing windows/files).

If there is not enough memory in reserve to cover a request, we notify the user of
impending doom. The Toolbox Memory Manager will return an error code of memFullErr.
This will probably cause some kind of System Bomb.

 *****************************************************************************/


long  CApplication::GrowMemory(
      Size  bytesNeeded)
{
      Size  bytesToFree;
      Size  rainyDayBalance;
      Size  grow;

      MemoryShortage(bytesNeeded);
                                          /* Try to release some memory       */
                                          /* See how much we still need       */
      bytesToFree = bytesNeeded - MaxMem(&grow);
      if (bytesToFree < 1)
                                          /* Hooray! Enough memory was freed! */
            return(GROW_SUCCESS);           /*   Let's get out of here          */
                                            /* Still need more. Dip into the    */
                                            /* rainy day fund if the request is */
                                            /* within the creditLimit or        */
                                            /*   if the loan has been approved. */
        if ((rainyDay != NULL) &&
            (loanApproved || bytesToFree <= creditLimit)) {

            rainyDayBalance = GetHandleSize(rainyDay);
            if (rainyDayBalance - bytesToFree >= MINIMUM_BALANCE) {

                                            /* We have enough savings to cover  */
                                            /*   the request and maintain our   */
                                            /*   minimum balance                */
                SetHandleSize(rainyDay, MINIMUM_BALANCE);

            } else if (rainyDayBalance >= bytesToFree) {
                                            /* We have enough to cover the      */
                                            /*   request but not enough to keep */
                                            /*   our minimum balance. Cash in   */
                                            /*   the entire rainy day fund.     */
                DisposHandle(rainyDay);
                rainyDay = NULL;

            } else {                        /* We don't have enough reserve to  */
                                            /*   cover the request. Liquidate   */
                                            /*   the entire fund and hope for   */
                                            /*   the best.                      */
                DisposHandle(rainyDay);
                rainyDay = NULL;
            }

            gSleepTime = 0L;                /* Idle as soon as possible         */
            rainyDayUsed = TRUE;            /* We've tapped the rainy day fund  */
            return(GROW_SUCCESS);           /* Exit here                        */

        } else {                            /* No more memory can be freed up   */

            if (canFail) {                  /* Requestor is prepared for        */
                return(GROW_FAILURE);       /*   failure                        */
            } else {                        /* Try the last resort              */
                return( OutOfMemory(bytesNeeded) );
            }
        }
}
```

```
/******************************************************************************

 MemoryShortage


There is not enough memory to fill a request. Try to free up some memory.
Subclasses should override this method.

The application should free all non-essential blocks of memory (such as those used
to cache data for speed or resources marked as unpurgeable to allow for quick
access) and disable commands which could consume a lot of memory (such as opening
files or creating new documents). The rainy day fund created by InitMemory should
be large enough to allow (or even force) the user to take actions to reduce the
memory strain without losing data (such as closing windows/files).

IMPORTANT: This method should NOT allocate any blocks of memory.


******************************************************************************/


void  CApplication::MemoryShortage(
      Size  bytesNeeded)
                                          /* Amount of memory requested        */
{
}                                         /* Null Method                       */


/******************************************************************************

 MemoryReplenished

The rainy day fund of reserve memory has been replenished. Subclasses must override
this method if certain actions taken by the MemoryShortage method need to be
undone. For example, if the "Open..." command was disabled by MemoryShortage, it
should be enabled here.
 ******************************************************************************/


void  CApplication::MemoryReplenished()
{
}                                         /* Null Method                       */



/******************************************************************************

 OutOfMemory


A memory request cannot be satisfied. At this point, the rainy day fund has been
liquidated. This method posts an Alert if there's enough memory to do so, and then
jumps back to the event loop.
```

```
Classes which override this method should pass back the appropriate return value
(same as that for a GrowZone function) if necessary. This method performs a long
jump, so no return value is needed.
 *************************************************************************/


long    CApplication::OutOfMemory(
        Size  bytesNeeded)                   /* Amount of memory requested        */
{
        Size  grow;

        if (MaxMem(&grow) > MINIMUM_BALANCE) {
              gError->CheckOSError(memFullErr);
        }

        HiliteGUI(NOTHING);
        JumpToEventLoop();
}



/**** E X E C U T I O N   M E T H O D S ****/

/************************************************************************


            Run the application by processing events.

 ************************************************************************/

            /*
             * Perform a Chore
             */
            static void Chore_Perform(
                  CChore       *theChore,
                  long   *minSleep)
            {
                  long   maxSleep = MAXLONG;

                  theChore->Perform(&maxSleep);
                  *minSleep = Min(*minSleep, maxSleep);
            }


void    CApplication::Run()
{
        register Boolean   aDAIsActive;
                                             /* Is a DA the front window?        */
        long                        minSleep = 0;

        Preload();                           /* Open/Print files selected from   */
                                             /* the W95/W98 Process before       */
                                             /* launching application            */
```

```
    f_SetJump(eventLoopJump);

    aDAIsActive = IsSystemWindow((WindowPeek)FrontWindow());
    do {
                                        /* Continuously poll for events     */
        itsSwitchboard->ProcessEvent();
                                        /*   and respond to them            */
        if (urgentsToDo) {
                                        /* Carry out urgent chores          */
            itsUrgentChores->DoForEach1(Chore_Perform, (long)&minSleep);
            itsUrgentChores->DisposeItems();
            urgentsToDo = FALSE;
        }

                                        /* Check for context-switch with    */
                                        /*   UNIX Accessories               */
        if (IsSystemWindow((WindowPeek)FrontWindow())) {
            gSleepTime = 0;
            if (!aDAIsActive) {
                aDAIsActive = TRUE;
                SwitchToDA();
            }
        } else {
            if (aDAIsActive) {
                aDAIsActive = FALSE;
                SwitchFromDA();
            }
        }

    } while (running);                   /*   Until flag is turned off      */
}


/*****************************************************************************
 Preload

Open/Print files selected by the user from the W95/W98 Mngr before launching the
application
 *****************************************************************************/


void  CApplication::Preload()
{

    short               openOrPrint;      /* Type of action requested        */
    short               numPreloads;      /* No. of files to process         */
    register short      i;                /* Index for files                 */
    AppFile                   UNIXAppFile;/* Info about a file               */
    SFReply                   UNIXSFReply;/* Standard File info record       */
                                          /* Check if files were selected    */
    CountAppFiles(&openOrPrint, &numPreloads);
    for (i = 1; i <= numPreloads; i++) {
                                        /* Process files one by one        */
        GetAppFiles(i, &UNIXAppFile);   /* Get info about the file         */
                                        /* Copy info into a Standard File  */
                                        /*   record                        */

        UNIXSFReply.fType = UNIXAppFile.fType;
        UNIXSFReply.vRefNum = UNIXAppFile.vRefNum;
        UNIXSFReply.version = UNIXAppFile.versNum;
        CopyPString(UNIXAppFile.fName, UNIXSFReply.fName);

        OpenDocument(&UNIXSFReply);    /* Automatically open selected file */

        if (openOrPrint == appPrint) {
                                        /* User wants to print Document    */

            gDesktop->UpdateWindows();
            gGopher->DoCommand(cmdPrint);
        }

        ClrAppFiles(i);                /* We've processed this file       */
    }

    StartUpAction(numPreloads);         /* Perform start up action         */
}


/*****************************************************************************

 StartUpAction

Perform any desired start up action. This method is invoked after files selected
from the W95/W98 Mngr have been opened/printed. The number of preloaded files is
passed as a parameter. If no files were preloaded, this default method acts as if
the user selected "New" and sends the Gopher a DoCommand(cmdNew) message.


*****************************************************************************/

void    CApplication::StartUpAction(
        short numPreloads)
{
        FlushEvents(everyEvent, 0);
        if (numPreloads == 0) {
            gGopher->DoCommand(cmdNew);
        }
}
/*****************************************************************************
 Suspend
Program is about to be swapped out under
/*****************************************************************************
 CApplication.c
```

```
                                            The Application Class
      Abstract class for a Mac application. Programmers must create a subclass
      of CApplication for their own programs. The application is at the top of
      the chain of command and defines the behavior of the program and its
      interaction with the Mac OS.

      SUPERCLASS = CBureaucrat
      Copyright © 1989 Symantec Corporation. All rights reserved.

            written by: Gregory H. Dow

 ******************************************************************************/

#include "Global.h"
#include "Constants.h"
#include "CApplication.h"
#include "Commands.h"
#include "OSChecks.h"
#include "CDirector.h"
#include "CError.h"
#include "TBUtilities.h"
#include "CDesktop.h"
#include "CBartender.h"
#include "CClipboard.h"
#include "CDecorator.h"

/*** Global Variables ***/

extern CApplication     *gApplication;     /* Application object              */
extern CDesktop         *gDesktop;         /* The visible Desktop             */
extern CBartender       *gBartender;       /* Manages all GUIs                */
extern CClipboard       *gClipboard;       /* Copies and Pastes data          */
extern CBureaucrat      *gGopher;          /* First in line to get commands   */
extern CError           *gError;           /* Error handler                   */
extern CDecorator       *gDecorator;       /* Decorator for arranging windows */
extern long             gSleepTime;        /* Max time between events         */
extern Boolean          gHasWNE;           /* Is WaitNextEvent implemented?   */
extern Boolean          gInBackground;     /* In background under MultiFinder */
extern OSType           gSignature;        /* Creator for Application's files */
extern EventRecord      gLastMouseDown;    /* Previous mouse down event       */
extern EventRecord      gLastMouseUp;      /* Previous mouse up event         */
extern CView            *gLastViewHit;     /* Last view clicked in            */
extern short            gClicks;           /* Click counter, = 1 single click */
                                           /*                = 2 double click */
                                           /*    etc.                         */
extern CursHandle gIBeamCursor;            /* I-beam for text views           */
extern CursHandle gWatchCursor;            /* Watch cursor for waiting        */
extern RgnHandle  gUtilRgn;                /* Utility region                  */

/*** Class Constants ***/

#define           GROW_FAILURE      0L     /* Return codes from GrowZoneFunc  */
#define           GROW_SUCCESS      1L
#define           JUMPBUFFER_A1     5      /* Index of A1 in JumpBuffer       */
```

```
/**** I N I T I A L I Z A T I O N   M E T H O D S ****/


/**************************************************************************
 IApplication

            Initialize an Application.
 **************************************************************************/

void  CApplication::IApplication(
      short      extraMasters,          /* Number of additional master      */
                                        /*   blocks to allocate             */
      Size       aRainyDayFund,         /* Bytes of memory to reserve for   */
                                        /*   a rainy day                    */
      Size       aCreditLimit)          /* Max unapproved memory request    */
{
                                        /* We haven't reached the event     */
                                        /* loop yet. Flag A1 (jump addr)    */
                                        /* so we don't try to jump there.   */
      eventLoopJump[JUMPBUFFER_A1] = NULL;

      InitToolbox();                    /* Initialize Toolbox Managers      */
                                        /* Fine tune the Memory Manager     */
      InitMemory(extraMasters, aRainyDayFund, aCreditLimit);

/** Instance Variables **/
      itsSwitchboard = new(CSwitchboard);
      itsSwitchboard->ISwitchboard();
      itsDirectors = new(CCluster);
      itsDirectors->ICluster();
      itsIdleChores = new(CList);
      itsIdleChores->IList();
      itsUrgentChores = new(CCluster);
      itsUrgentChores->ICluster();
      urgentsToDo = FALSE;
      running = TRUE;

                                        /** Global Variables **/
      gSignature = '????';
      gHasWNE = WNEIsImplemented();
      gSleepTime = 0;                   /* We want an early first Idle      */
      gError = new(CError);             /* Cursors                          */
      gIBeamCursor = GetCursor(iBeamCursor);
      HNoPurge(gIBeamCursor);
      gWatchCursor = GetCursor(watchCursor);
      HNoPurge(gWatchCursor);


      gUtilRgn = NewRgn();
      MakeDesktop();
      MakeClipboard();
      MakeDecorator();
      SetUpFileParameters();
      SetUpGUIs();
      gGopher = this;
      gLastViewHit = NULL;
      gLastMouseUp.when = 0L;
      gClicks = 0;
}


/*******************************************************************************
 InitToolbox

           Initialize the Macintosh Toolbox
 *******************************************************************************/

void  CApplication::InitToolbox()
{
      InitGraf(&thePort);                  /* Standard initialization calls    */
      InitFonts();
      InitWindows();
      InitGUIs();
      TEInit();
      InitDialogs(NULL);                   /* ??? Add a ResumeProc             */
      InitCursor();
}


/*******************************************************************************
 InitMemory

           Initialize the Heap and create extra master pointer blocks
 *******************************************************************************/

void  CApplication::InitMemory(
      short       extraMasters,            /* Number of additional master      */
                                           /* blocks to allocate               */
      Size        aRainyDayFund,           /* Bytes of memory to reserve for   */
                                           /* a rainy day                      */
      Size        aCreditLimit)            /* Max memory request which can be  */
                                           /* taken from rainyDayFund          */
                                           /* without prior approval           */
{
      MaxApplZone();                       /* Grow heap to maximum size to     */
                                           /* help prevent fragmentation       */

                  /* Call MoreMasters to allocate extra master pointers.  */
                  /* Experiment to determine how many master pointers the */
                  /* application uses under session of heavy use.      */

      while (extraMasters-- > 0)
            MoreMasters();

      SetGrowZone(GrowZoneFunc);           /* Traps out of memory conditions   */
      rainyDay = NULL;                     /* Clear rainyDay handle in case    */
                                           /*   allocating the rainy day fund  */
                                           /*   fail! This is really bad.      */
	rainyDayFund = aRainyDayFund;		/* Set instance variables		*/
	creditLimit = aCreditLimit;
	rainyDayUsed = FALSE;
	memWarningIssued = FALSE;
	loanApproved = FALSE;

	rainyDay = NewHandle(rainyDayFund); /* Reserve a block of memory	*/
}


/******************************************************************************
 MakeDesktop

		Create the global Desktop object, which is the top level of
		the visual hierarchy. Overrride this method to use a desktop
		which supports floating windows or other extensions to the
		standard desktop.
 ******************************************************************************/

void	CApplication::MakeDesktop()
{
	gDesktop = new(CDesktop);		/* Use a standard Desktop		*/
	gDesktop->IDesktop(this);
}

/******************************************************************************
 MakeClipboard

		Create the global Clipboard object
 ******************************************************************************/

void	CApplication::MakeClipboard()
{
	gClipboard = new(CClipboard);		/* Use standard Clipboard		*/
	gClipboard->IClipboard(this, TRUE);
}

/******************************************************************************
 MakeDecorator

		Create the global Decorator object, which controls the
		arrangement of windows on the screen. Override if a
		different Decorator is desired.
 ******************************************************************************/

void	CApplication::MakeDecorator()
{
	gDecorator = new(CDecorator);		/* Use standard Decorator		*/
	gDecorator->IDecorator();
}
```

```
/*************************************************************************
 SetUpFileParameters

           Set values of the parameters used by the Standard File Package.
 *************************************************************************/

void  CApplication::SetUpFileParameters()
{

/** Get File Parameters **/
      sfNumTypes = -1;                    /* Display all files             */
      sfFileTypes[0] = '????';            /* Just put something here       */
      sfFileFilter = NULL;                /* No extra filtering of files   */
      sfGetDLOGHook = NULL;               /* No special handling of items  */
      sfGetDLOGid = getDlgID;             /* Use built-in get dialog box   */
      sfGetDLOGFilter = NULL;             /* No special event processing   */
}


/*************************************************************************
 SetUpGUIs

           Set up the GUIs used by the application
 *************************************************************************/

void  CApplication::SetUpGUIs()
{
      gBartender = new(CBartender);
      gBartender->IBartender(MBARapp);
      AddResGUI(GetMHandle(GUIVector), 'DRVR');
      gBartender->SetDimOption(GUIVector, dimNONE);
}



/**** A C T I O N   M E T H O D S ****/


/*************************************************************************
 Notify {OVERRIDE}

           A subordinate is notifying the Application that a task has been
           completed.
 *************************************************************************/

void  CApplication::Notify(
      CTask       *theTask)               /* The completed task            */
{
}                                         /* Null Method                   */
```

```
/***************************************************************************
 DoKeyDown {OVERRIDE}

            Respond to a keystroke. Application has no supervisor to whom to
            pass the buck. Default action is to ignore keystrokes.
 ***************************************************************************/

void  CApplication::DoKeyDown(
      char            theChar,          /* The associated character       */
      Byte            keyCode,          /* Code for the associated key    */
      EventRecord     *macEvent)        /* Key down event record          */
{
}                                       /* Null Method                    */

/***************************************************************************
 DoAutoKey {OVERRIDE}

            Respond to a key being held down. By default, keystrokes are ignored.
 ***************************************************************************/

void  CApplication::DoAutoKey(
      char            theChar,          /* The associated character       */
      Byte            keyCode,          /* Code for the associated key    */
      EventRecord     *macEvent)        /* Autokey event record           */
{
}                                       /* Null Method                    */

/***************************************************************************
 DoKeyUp {OVERRIDE}

            Respond to a key being released. Usually, these events will be
            masked out by the operating system. Ignore them by default.
 ***************************************************************************/

void  CApplication::DoKeyUp(
      char            theChar,          /* The associated character       */
      Byte            keyCode,          /* Code for the associated key    */
      EventRecord     *macEvent)        /* Key up event record            */
{
}                                       /* Null Method                    */


/***************************************************************************
 DoCommand

            Handle a command. Application has no supervisor which can handle the
            command. If we don't handle the command here, it will never be
            executed (and that's an error).
 ***************************************************************************/
```

```
void   CApplication::DoCommand(
       register long      theCommand)        /* Command number                    */
{
       Str255             theDA;             /* Name of Desk Accessory to open    */
       SFReply            macSFReply;        /* Standard File reply record        */

       if (theCommand < 0) {
             if (HiShort(-theCommand) == GUIVector) {
                                             /* Open a DA or launch another       */
                                             /*   application under MultiFinder   */
                   GetItem(GetMHandle(GUIVector), LoShort(-theCommand), theDA);
                   OpenDeskAcc(theDA);
             }
       } else {

             switch (theCommand) {

                   case cmdNew:
                         SetCursor(*gWatchCursor);
                         CreateDocument();
                         break;

                   case cmdOpen:
                         ChooseFile(&macSFReply);
                         if (macSFReply.good) {
                               SetCursor(*gWatchCursor);
                               OpenDocument(&macSFReply);
                         }
                         break;

                   case cmdClose:
                         if (IsSystemWindow((WindowPeek)FrontWindow()))
                               CloseDeskAcc(((WindowPeek)FrontWindow())-windowKind);
                         break;

                   case cmdQuit:
                         Quit();
                         break;

                   case cmdUndo:
                   case cmdCut:
                   case cmdCopy:
                   case cmdPaste:
                   case cmdClear:
                         SystemEdit((short)(theCommand - cmdUndo));
                         break;

                   case cmdToggleClip:
                         gClipboard->Toggle();
                         break;
             }
       }
}
```

```
/***************************************************************************
 UpdateGUIs {OVERRIDE}

            Perform GUI management tasks
 ***************************************************************************/

void  CApplication::UpdateGUIs()
{
      Str63 undoStr;

      gBartender->EnableCmd(cmdQuit);

      if (gInBackground) {
            gBartender->EnableCmd(cmdClose);
            gBartender->EnableCmd(cmdUndo);
            gBartender->EnableCmd(cmdCut);
            gBartender->EnableCmd(cmdCopy);
            gBartender->EnableCmd(cmdPaste);
            gBartender->EnableCmd(cmdClear);
      } else {
            gBartender->EnableCmd(cmdToggleClip);
      }

      if (!rainyDayUsed) {
            gBartender->EnableCmd(cmdNew);
            gBartender->EnableCmd(cmdOpen);
      }

      GetIndString(undoStr, STRcommon, strUNDO);
      gBartender->SetCmdText(cmdUndo, undoStr);
}


/**** M E M O R Y   M A N A G E M E N T ****/


/***************************************************************************
 RequestMemory

            Turn on/off the flags which indicate whether subsequent memory
            requests can use the rainy day fund and can fail
 ***************************************************************************/

void  CApplication::RequestMemory(
      Boolean            aLoanApproved,
      Boolean            aCanFail)
{
      loanApproved = aLoanApproved;
      canFail = aCanFail;
}
```

```
/************************************************************************
 GrowMemory

           A memory request can't be filled by the Mac Memory Manager. Free up
           memory if possible or gracefully shutdown.

           A rainy day fund of reserve memory is maintained to monitor low
           memory conditions. If the application can't free up enough memory
           on its own, we use the reserve fund. The user is notified when
           we tap into the reserve fund so that he/she can take steps to free
           up memory (perhaps by closing windows/files).

           If there is not enough memory in reserve to cover a request, we
           notify the user of impending doom. The Toolbox Memory Manager will
           return an error code of memFullErr. This will probably cause some kind
           of System Bomb.

 ************************************************************************/

long  CApplication::GrowMemory(
      Size  bytesNeeded)
{
      Size  bytesToFree;
      Size  rainyDayBalance;
      Size  grow;

      MemoryShortage(bytesNeeded);        /* Try to release some memory       */
                                          /* See how much we still need       */
      bytesToFree = bytesNeeded - MaxMem(&grow);

      if (bytesToFree < 1)                /* Hooray! Enough memory was freed! */
            return(GROW_SUCCESS);         /* Let's get out of here            */
                                          /* Still need more. Dip into the    */
                                          /*   rainy day fund if the request  */
                                          /*   is within the creditLimit or   */
                                          /*   if the loan has been approved. */
      if ((rainyDay != NULL) &&
            (loanApproved || bytesToFree <= creditLimit)) {


            rainyDayBalance = GetHandleSize(rainyDay);
            if (rainyDayBalance - bytesToFree >= MINIMUM_BALANCE) {
                                          /* We have enough savings to cover  */
                                          /*   the request and maintain our   */
                                          /*   minimum balance                */
                  SetHandleSize(rainyDay, MINIMUM_BALANCE);

            } else if (rainyDayBalance >= bytesToFree) {
                                          /* We have enough to cover the      */
                                          /*   request but not enough to keep */
                                          /*   our minimum balance. Cash in   */
                                          /*   the entire rainy day fund.     */
                  DisposHandle(rainyDay);
                  rainyDay = NULL;
            } else {                        /* We don't have enough reserve to  */
                                            /*   cover the request. Liquidate   */
                                            /*   the entire fund and hope for   */
                                            /*   the best.                      */
                  DisposHandle(rainyDay);
                  rainyDay = NULL;
            }

            gSleepTime = 0L;                /* Idle as soon as possible         */
            rainyDayUsed = TRUE;            /* We've tapped the rainy day fund  */
            return(GROW_SUCCESS);           /* Exit here                        */
      } else {                              /* No more memory can be freed up   */

            if (canFail) {                  /* Requestor is prepared for        */
                  return(GROW_FAILURE);     /*   failure
            */

            } else {                        /* Try the last resort              */
                  return( OutOfMemory(bytesNeeded) );
            }
      }
}


/*****************************************************************************
 MemoryShortage

            There is not enough memory to fill a request. Try to free up some
            memory. Subclasses should override this method.

            The application should free all non-essential blocks of
            memory (such as those used to cache data for speed or resources
            marked as unpurgeable to allow for quick access) and disable
            commands which could consume a lot of memory (such as opening files
            or creating new documents). The rainy day fund created by InitMemory
            should be large enough to allow (or even force) the user to take
            actions to reduce the memory strain without losing data (such as
            closing windows/files).

            IMPORTANT: This method should NOT allocate any blocks of memory.
 *****************************************************************************/


void  CApplication::MemoryShortage(
      Size  bytesNeeded)                    /* Amount of memory requested       */
{
}                                           /* Null Method                      */
```

```
/*****************************************************************************
 MemoryReplenished

            The rainy day fund of reserve memory has been replenished.
            Subclasses must override this method if certain actions taken
            by the MemoryShortage method need to be undone. For example, if
            the "Open..." command was disabled by MemoryShortage, it should
            be enabled here.
 *****************************************************************************/

void  CApplication::MemoryReplenished()
{
}                                         /* Null Method                    */


/*****************************************************************************
 OutOfMemory

            A memory request cannot be satisfied. At this point, the rainy day
            fund has been liquidated. This method posts an Alert if there's
            enough memory to do so, and then jumps back to the event loop.
            Classes which override this method should pass back the appropriate
            return value (same as that for a GrowZone function) if necessary.
            This method performs a long jump, so no return value is needed.
 *****************************************************************************/

long  CApplication::OutOfMemory(
      Size  bytesNeeded)                  /* Amount of memory requested       */
{
      Size  grow;

      if (MaxMem(&grow) > MINIMUM_BALANCE) {
            gError->CheckOSError(memFullErr);
      }

      HiliteGUI(NOTHING);
      JumpToEventLoop();
}


/**** E X E C U T I O N   M E T H O D S ****/


/*****************************************************************************
 Run

            Run the application by processing events.
 *****************************************************************************/

            /*
             * Perform a Chore
             */
            static void Chore_Perform(
                  CChore      *theChore,
                  long  *minSleep)
            {
                  long  maxSleep = MAXLONG;

                  theChore->Perform(&maxSleep);
                  *minSleep = Min(*minSleep, maxSleep);
            }

void  CApplication::Run()
{
      register Boolean  aDAIsActive;      /* Is a DA the front window?        */
      long                    minSleep = 0;

      Preload();                          /* Open/Print files selected from   */
                                          /*   the Finder before launching    */
                                          /*   application                    */

      f_SetJump(eventLoopJump);

      aDAIsActive = IsSystemWindow((WindowPeek)FrontWindow());
      do {                                /* Continuously poll for events     */
            itsSwitchboard->ProcessEvent(); /*   and respond to them          */

            if (urgentsToDo) {            /* Carry out urgent chores          */
                  itsUrgentChores->DoForEach1(Chore_Perform, (long)&minSleep);
                  itsUrgentChores->DisposeItems();
                  urgentsToDo = FALSE;
            }

                                          /* Check for context-switch with    */
                                          /*   Desk Accessories               */
            if (IsSystemWindow((WindowPeek)FrontWindow())) {
                  gSleepTime = 0;
                  if (!aDAIsActive) {
                        aDAIsActive = TRUE;
                        SwitchToDA();
                  }
            } else {
                  if (aDAIsActive) {
                        aDAIsActive = FALSE;
                        SwitchFromDA();
                  }
            }

      } while (running);                  /*   Until flag is turned off       */
}


/********************************************************************************
 Preload

            Open/Print files selected by the user from the Finder before
            launching the application
 ********************************************************************************/

void  CApplication::Preload()
```

```
{
      short             openOrPrint;       /* Type of action requested         */
      short             numPreloads;       /* No. of files to process          */
      register short    i;                 /* Index for files                  */
      AppFile                 macAppFile;  /* Info about a file                */
      SFReply                 macSFReply;  /* Standard File info record        */
                                           /* Check if files were selected     */
      CountAppFiles(&openOrPrint, &numPreloads);

      for (i = 1; i <= numPreloads; i++) {/* Process files one by one          */
            GetAppFiles(i, &macAppFile);   /* Get info about the file          */
                                           /* Copy info into a Standard File   */
                                           /* record                           */
            macSFReply.fType = macAppFile.fType;
            macSFReply.vRefNum = macAppFile.vRefNum;
            macSFReply.version = macAppFile.versNum;
            CopyPString(macAppFile.fName, macSFReply.fName);

            OpenDocument(&macSFReply);     /* Automatically open selected file */

            if (openOrPrint == appPrint) { /* User wants to print Document     */
                  gDesktop->UpdateWindows();
                  gGopher->DoCommand(cmdPrint);
            }

            ClrAppFiles(i);                /* We've processed this file        */
      }

      StartUpAction(numPreloads);          /* Perform start up action          */
}

/******************************************************************************
 StartUpAction

            Perform any desired start up action. This method is invoked after
            files selected from the Finder have been opened/printed. The number
            of preloaded files is passed as a parameter. If no files were
            preloaded, this default method acts as if the user selected "New"
            and sends the Gopher a DoCommand(cmdNew) message.
 ******************************************************************************/

void  CApplication::StartUpAction(
      short numPreloads)
{
      FlushEvents(everyEvent, 0);
      if (numPreloads == 0) {
            gGopher->DoCommand(cmdNew);
      }
}
```

```
/*****************************************************************************
 Suspend

            Program is about to be swapped out under MultiFinder. Send the
            Suspend message to each of the Application's Directors.
 *****************************************************************************/

            /*
             *    Send Suspend message to a director
             */
                  static void Director_Suspend(
                        CDirector   *theDirector)
                  {
                  theDirector->Suspend();
                  }

void  CApplication::Suspend()
{
                                           /* Tell all our directors that we   */
                                           /*   are about to lose control      */
      itsDirectors->DoForEach(Director_Suspend);
      gInBackground = TRUE;                /* We are going into the background */
}

/*****************************************************************************
 Resume

            Program is being swapped in under MultiFinder
 *****************************************************************************/

            /*
             *    Send Resume message to a director
             */
                  static void Director_Resume(
                        CDirector   *theDirector)
                  {
                        theDirector->Resume();
                  }

void  CApplication::Resume()
{
                                           /* Do nothing if a DA is in front   */
      if (!IsSystemWindow((WindowPeek)FrontWindow())) {
            gInBackground = FALSE;         /* We are no longer in background   */
                                           /* Tell all directors we are back   */
                                           /*   in control                     */
            itsDirectors->DoForEach(Director_Resume);
      }
}
/*****************************************************************************
 SwitchToDA

            A DA is becoming active
 *****************************************************************************/
```

```
void  CApplication::SwitchToDA()
{
      CurDeactive = NULL;
      Suspend();
}


/******************************************************************************
 SwitchFromDA

            Our application is becoming active after a DA had been running
 ******************************************************************************/

void  CApplication::SwitchFromDA()
{
      CurActivate = NULL;
      Resume();
}


/******************************************************************************
 Quit

            User requested quit. Need to perform a graceful shutdown,
            giving the user the opportunity to save changed files or to abort
            the quit process.
 ******************************************************************************/

void  CApplication::Quit()
{
      register CWindow  *theWindow;
      CDirector               *theDirector;

      running = FALSE;                    /* Assume we will stop running      */
                                          /* Run through all our windows      */
                                          /*   (Does not include floating     */
                                          /*    windows or DAs)               */

      theWindow = gDesktop->GetTopWindow();
      while (theWindow != NULL) {
            theDirector = (CDirector*) theWindow->GetSupervisor();
                                          /* Tell window's Director to quit   */
                                          /*   If Quit methode returns FALSE, */
                                          /*   abort the quit sequence        */
            if (!theDirector->Quit()) {
                  running = TRUE;         /* User still wants to run program  */
                  break;
            }
            theWindow = gDesktop->GetTopWindow();
      }
}
```

```
/*****************************************************************************
 Exit

            Program is about to terminate. Last chance to clean up.
 *****************************************************************************/

void  CApplication::Exit()
{
}                                           /* NULL Method                  */


/*****************************************************************************
 JumpToEventLoop

            Jump to the beginning of the main event loop. Useful when trying
            to recover from an error.

            An error may occur BEFORE program ever gets to the event loop. In
            this case the jump buffer entry for A1 (the address to jump to) will
            be NULL and we exit the program.
 *****************************************************************************/

void  CApplication::JumpToEventLoop()
{
      if (eventLoopJump[JUMPBUFFER_A1] != NULL) {
            f_LongJump(eventLoopJump, 1);

      } else {                              /* We never got to the event loop   */
            Exit();                         /* Not much we can do except abort  */
            ExitToShell();                  /* the program                      */
      }
}


/**** D O C U M E N T   H A N D L I N G   M E T H O D S ****/

/*****************************************************************************
 CreateDocument

            Make a document in response to the "New" GUI selection. This method
            must be overridden if the standard "New" command is used.
 *****************************************************************************/

void  CApplication::CreateDocument()
{
}                                           /* NULL Method                  */


/*****************************************************************************
 OpenDocument

            Open an existing file and create a document object for displaying
            information. This method must be overridden if the standard "Open"
            command is used.
 *****************************************************************************/
```

```
void  CApplication::OpenDocument(
      SFReply            *macReply)
{
}                                          /* NULL Method                   */


/***************************************************************************
 ChooseFile

            Let the user select a file to open. Default procedure is to use
            the Standard GetFile dialog box.
 **************************************************************************/

void  CApplication::ChooseFile(
      SFReply            *macSFReply)
{
      Point        corner;                 /* Top left corner of dialog box   */
      SignedByte   saveHState;

                                           /* Center dialog box on the screen  */
      FindDlogPosition('DLOG', sfGetDLOGid, &corner);

      saveHState = HGetState(this);
      HLock(this);

      SFPGetFile(corner, NULL, sfFileFilter, sfNumTypes, sfFileTypes,
                        sfGetDLOGHook, macSFReply, sfGetDLOGid, sfGetDLOGFilter);

      HSetState(this, saveHState);
}


/***************************************************************************
 AddDirector

            Add a Director to an Application's list of Directors
 **************************************************************************/

void  CApplication::AddDirector(
      CDirector    *aDirector)             /* Director to add               */
{
      itsDirectors->Add(aDirector);
}


/***************************************************************************
 RemoveDirector

            Eliminate a Director
 **************************************************************************/

void  CApplication::RemoveDirector(
      CDirector    *theDirector)           /* Director to remove            */
{
      itsDirectors->Remove(theDirector);
}


/**** C H O R E   H A N D L I N G   M E T H O D S ****/


/*******************************************************************************
 Idle

            No event directed to our application is pending. Here's our chance
            to perform periodic tasks. The classic example is flashing a
            text insertion cursor.
 ******************************************************************************/

void  CApplication::Idle(
      EventRecord        *macEvent)        /* Usually a null or system event   */
{
      Size                          grow; /* Amount heap may be grown          */
      register CBureaucrat    *theBureaucrat;
      long                          minSleep;
      long                          maxSleep;


      if (rainyDayUsed) {                  /* Memory reserve needs to be       */
                                           /*   replenished                    */
            if (MaxMem(&grow) > rainyDayFund + MINIMUM_BALANCE) {
                                           /* Enough memory has been freed     */
                  if (rainyDay != NULL) {
                        DisposHandle(rainyDay);
                  }
                  rainyDay = NewHandle(rainyDayFund);
                  rainyDayUsed = FALSE;
                  memWarningIssued = FALSE;
                  MemoryReplenished();
            } else if (!memWarningIssued) {
                                           /* Inform user of memory shortage   */
                  gError->PostAlert(STRmemWarn, iMEM_LOW);
                  memWarningIssued = TRUE;
            }
      }


                                           /* Idle time for objects in the     */
                                           /*   active chain of command        */
      theBureaucrat = gGopher;             /* Start at the bottom              */
      minSleep = MAXLONG;
      do {                                 /* Run thru the chain of command    */
                                           /*   Let bureaucrat kill some time  */
            maxSleep = MAXLONG;
            theBureaucrat->Dawdle(&maxSleep);
            minSleep = Min(minSleep, maxSleep);
                                           /*   Move up to his/her boss        */
            theBureaucrat = theBureaucrat->itsSupervisor;
      } while (theBureaucrat != NULL);     /* Until end of command chain       */
```

```
                                            /* Carry out assigned chores        */
     itsIdleChores->DoForEach1(Chore_Perform, (long)&minSleep);
     gSleepTime = minSleep;
}


/******************************************************************************
 AssignIdleChore

          Give Application a chore to perform at idle time. Idle chores are
          performed whenever there are no events pending for the Application.
 ******************************************************************************/

void  CApplication::AssignIdleChore(
     CChore      *theChore)                 /* Chore to perform                 */
{
     itsIdleChores->Add(theChore);
     gSleepTime = 0;                        /* Force immediate idle             */
}


/******************************************************************************
 CancelIdleChore

          Relieve Application of an idle time chore. A chore must NOT cancel
          itself or any other chore.
 ******************************************************************************/

void  CApplication::CancelIdleChore(
     CChore      *theChore)                 /* Chore to cancel                  */
{
     itsIdleChores->Remove(theChore);
}


/******************************************************************************
 AssignUrgentChore

          Give Application a chore to perform as soon as possible. Urgent
          chores are performed only once, then automatically disposed.
          An urgent chore must NOT post another urgent chore.
 ******************************************************************************/

void  CApplication::AssignUrgentChore(
     CChore      *theChore)                 /* Chore to perform                 */
{
     itsUrgentChores->Add(theChore);
     urgentsToDo = TRUE;
}
. Send the Suspend message to each of the Application's Directors.
 ******************************************************************************/
          /*
           *    Send Suspend message to a director
           */
               static void Director_Suspend(
                        CDirector    *theDirector)
                {
                theDirector->Suspend();
                }

void  CApplication::Suspend()
{
                                        /* Tell all our directors that we   */
                                        /*   are about to lose control      */

    itsDirectors->DoForEach(Director_Suspend);
    gInBackground = TRUE;               /* We are going into the background */
}


/*****************************************************************************
 Resume

         Program is being swapped in under UNIX /temp
 *****************************************************************************/

         /*
          *    Send Resume message to a director
          */
              static void Director_Resume(
                    CDirector    *theDirector)
              {
                    theDirector->Resume();
              }


void  CApplication::Resume()
{
                                        /* Do nothing if a W95/W98 PROCESS  */
                                        /* is in front                      */
    if (!IsSystemWindow((WindowPeek)FrontWindow())) {
         gInBackground = FALSE;
                                        /* We are no longer in background   */
                                        /* Tell all directors we are back   */
                                        /*   in control                     */
         itsDirectors->DoForEach(Director_Resume);
    }
}



/*****************************************************************************
 SwitchToW95/W98 PROCESS

A W95/W98 PROCESS is becoming active
 *****************************************************************************/
```

```
void  CApplication::SwitchToW95/W98 PROCESS()
{
      CurDeactive = NULL;
      Suspend();
}


/*************************************************************************
 SwitchFromW95/W98 PROCESS

Our application is becoming active after a W95/W98 PROCESS had been running
 *************************************************************************/

void  CApplication::SwitchFromW95/W98 PROCESS()
{
      CurActivate = NULL;
      Resume();
}


/*************************************************************************
 Quit

User requested quit. Need to perform a graceful shutdown, giving the user the
opportunity to save changed files or to abort the quit process.
 *************************************************************************/


void  CApplication::Quit()
{
      register CWindow        *theWindow;
      CDirector               *theDirector;

      running = FALSE;
                                          /*  Assume we will stop running     */
                                          /* Run through all our windows      */
                                          /* (Does not include floating       */
                                          /*    windows or W95/W98 PROCESSs)  */
      theWindow = gDesktop->GetTopWindow();
      while (theWindow != NULL) {
            theDirector = (CDirector*) theWindow->GetSupervisor();

                                          /* Tell window's Director to quit   */
                                          /*   If Quit methode returns FALSE, */
                                          /*   abort the quit sequence        */
            if (!theDirector->Quit()) {
                  running = TRUE;         /* User still wants to run program  */
                  break;
            }
            theWindow = gDesktop->GetTopWindow();
      }
}
```

```
/*****************************************************************************
 Exit

            Program is about to terminate. Last chance to clean up.

*****************************************************************************/

void  CApplication::Exit()
{
}           /* NULL Method */

/*****************************************************************************
 JumpToEventLoop

Jump to the beginning of the main event loop. Useful when trying to recover from an
error.

An error may occur BEFORE program ever gets to the event loop. In this case the
jump buffer entry for A1 (the address to jump to) will be NULL and we exit the
program.
 ****************************************************************************/

void  CApplication::JumpToEventLoop()
{
      if (eventLoopJump[JUMPBUFFER_A1] != NULL) {
            f_LongJump(eventLoopJump, 1);

      } else {
                                             /* We never got to the event loop   */
            Exit();
                                             /* Not much we can do except abort  */
            ExitToShell();                   /*   the program  */
      }
}



/**** I M A G E   H A N D L I N G   M E T H O D S ****/

/****************************************************************************
 CreateDocument

Make a document in response to the "New" GUI selection. This method must be
overridden if the standard "New" command is used.

****************************************************************************/

void  CApplication::CreateDocument()
```

```
{
}                                          /* NULL Method                      */

/******************************************************************************
 OpenDocument

Open an existing file and create a document object for displaying information.

This method must be overridden if the standard "Open"command is used.

******************************************************************************/


void  CApplication::OpenDocument(
      SFReply           *UNIXReply)
{
}                                          /* NULL Method                      */


/******************************************************************************
 ChooseFile

Let the user select a file to open. Default procedure is to use the Standard
GetFile dialog box.

******************************************************************************/

void  CApplication::ChooseFile(
      SFReply           *UNIXSFReply)
{
      Point       corner;                  /* Top left corner of               */
                                           /* dialog box                       */
      SignedByte  saveHState;              /* Center dialog box on the         */
                                           /* screen                           */
      FindDlogPosition('DLOG', sfGetDLOGid, &corner);

      saveHState = HGetState(this);
      HLock(this);

      SFPGetFile(corner, NULL, sfFileFilter, sfNumTypes, sfFileTypes,
            sfGetDLOGHook, UNIXSFReply, sfGetDLOGid, sfGetDLOGFilter);

      HSetState(this, saveHState);
}

/******************************************************************************
 AddDirector

            Add a Director to an Application's list of Directors
 ******************************************************************************/
```

```
void  CApplication::AddDirector(
      CDirector    *aDirector)              /* Director to add              */
{
      itsDirectors->Add(aDirector);
}


/*****************************************************************************
 RemoveDirector

 Eliminate a Director
 *****************************************************************************/

void  CApplication::RemoveDirector(
      CDirector    *theDirector)            /* Director to remove           */
{
      itsDirectors->Remove(theDirector);
}



/****  U N I X  -  C H O R E   H A N D L I N G   M E T H O D S ****/


/*****************************************************************************
 Idle


No event directed to our application is pending. Here's our chance to perform
periodic tasks. The classic example is flashing a text insertion cursor.

*****************************************************************************/

void  CApplication::Idle(
      EventRecord       *UNIXEvent)        /* Usually a null or system event  */

{
      Size                         grow; /* Amount heap may be grown        */

      register CBureaucrat    *theBureaucrat;
      long                         minSleep;
      long                         maxSleep;


      if (rainyDayUsed) {
                                         /* Memory reserve needs to be      */
                                         /*   replenished                   */
            if (MaxMem(&grow) > rainyDayFund + MINIMUM_BALANCE) {


                                         /* Enough memory has been freed    */
                  if (rainyDay != NULL) {
                        DisposHandle(rainyDay);
```

```
                }
                rainyDay = NewHandle(rainyDayFund);
                rainyDayUsed = FALSE;
                memWarningIssued = FALSE;
                MemoryReplenished();
            } else if (!memWarningIssued) {

                                        /* Inform user of memory shortage   */

                gError->PostAlert(STRmemWarn, iMEM_LOW);
                memWarningIssued = TRUE;
            }
        }


                                        /* Idle time for objects in the     */
                                        /*   active chain of command        */

        theBureaucrat = gGopher;

                                        /* Start at the bottom              */
        minSleep = MAXLONG;
        do {                            /* Run thru the chain of command    */
                                        /*   Let bureaucrat kill some time  */
            maxSleep = MAXLONG;
            theBureaucrat->Dawdle(&maxSleep);
            minSleep = Min(minSleep, maxSleep);
                                        /*   Move up to his/her boss        */
            theBureaucrat = theBureaucrat->itsSupervisor;
        } while (theBureaucrat != NULL);

                                        /* Until end of command chain       */
                                        /* Carry out assigned chores        */

        itsIdleChores->DoForEach1(Chore_Perform, (long)&minSleep);
        gSleepTime = minSleep;
}




/******************************************************************************
 AssignIdleChore


Give Application a chore to perform at idle time. Idle chores are performed
whenever there are no events pending for the Application.


******************************************************************************/

void  CApplication::AssignIdleChore(
      CChore      *theChore)
```

```
                                              /* Vector Chore to perform         */
{
     itsIdleChores->Add(theChore);
     gSleepTime = 0;                          /* Force immediate idle            */
}


/**********************************************************************
 CancelIdleChore


Relieve Application of an idle time chore. A chore must NOT cancel itself or any
other chore.

**********************************************************************/


void  CApplication::CancelIdleChore(
      CChore      *theChore)                   /* Chore to cancel                 */
{
      itsIdleChores->Remove(theChore);
}



/**********************************************************************
 AssignUrgentChore


Give Application a chore to perform as soon as possible. Urgent chores are
performed only once, then automatically disposed. An urgent chore must NOT post
another urgent chore.
 **********************************************************************/


void  CApplication::AssignUrgentChore(
      CChore      *theChore)                  /* Chore to perform                */
{
      itsUrgentChores->Add(theChore);
      urgentsToDo = TRUE;
}
```

Annex to

Additional Interim Report 3a (01 September 1998)
Conditional Estimation of Vector Patterns in Remote Sensing and GIS

contract no. N 68171 97 C 9027
contractor: UvA, Applied Logic Labratory/CCSOM
Principal Investigator: Dr. M. Masuch

1. Statement showing amount of unused funds at the end of the covered period

2nd Incrementally Funded Peric total December 1998 –September 1999	$	150,000.00
3rd Incrementally Funded Perio total October 1999 - July 2000	$	150,000.00
Total unused funds from Sept 1998 until Februari 2000:	$	300,000

2. List of impotant property acquired with contract funds during this period

none